Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/GB05/000969

International filing date: 14 March 2005 (14.03.2005)

Document type: Certified copy of priority document

Document details: Country/Office: GB
Number: 0407336.7
Filing date: 31 March 2004 (31.03.2004)

Date of receipt at the International Bureau: 21 April 2005 (21.04.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

PCT/GB2005/000969

INVESTOR IN PEOPLE

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed

Dated 7 April 2005

THE PATENT OFFICE
A
31 MAR 2004
LONDON

The Patent Office

Patents Act 1977
(Rule 16)

01APR04 E885544-1 D03052
P01/7700 0.00-0407336.7 ACCOUNT PWA

The Patent Office
Cardiff Road
Newport
Gwent NP10 8QQ

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

1. Your reference

A30471

2. Patent application number
(The Patent Office will fill in this part)

0407336.7

31 MAR 2004

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*

BRITISH TELECOMMUNICATIONS public limited company
81 NEWGATE STREET
LONDON, EC1A 7AJ, England
Registered in England: 1800000

Patents ADP number *(if you know it)*

1867002

If the applicant is a corporate body, give the country/state of its incorporation

UNITED KINGDOM

4. Title of the invention

PATHFINDING SYSTEM

5. Name of your agent *(if you have one)*

"Address for Service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*

BT GROUP LEGAL
INTELLECTUAL PROPERTY DEPARTMENT
PPC5A, BT CENTRE
81 NEWGATE STREET
LONDON, EC1A 7AJ, ENGLAND

Patents ADP number *(if you know it)*

~~1867001~~ 8772378001

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number

Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application	Date of filing *(day/month/year)*

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*

a) any applicant named in part 3 is not an inventor, or
b) there is an inventor who is not named as an applicant, or
c) any named applicant is a corporate body.

(See note (d))

YES

9. Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

Continuation sheets of this form	-	
Description	-	60
Claim(s)	-	4
Abstract	-	1
Drawing(s)	-	25 + 25

10. If you are also filing any of the following, state how may against each item

Priority Documents	
Translations of priority documents	
Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*	
Request for preliminary examination and search *(Patents Form 9/77)*	YES (1)
Request for substantive examination *(Patents Form 10/77)*	
Any other documents *(please specify)*	

11. e I/We request the grant of a patent on the basis of this application.

Signature(s) Date:

Simon Williamson **31 March 2004**

WILLIAMSON, Simeon Paul, Authorised Signatory

12. Name and daytime telephone number of person to contact in the United Kingdom

Mark WATSON **020 7356 6163**

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

a) If you need help to fill in this form or you have any questions, please contact the Patent Office on 0645 500505.

b) Write your answers in capital letters using black ink or you may type them.

c) If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.

d) If you have answered 'Yes' Patents Form 7/77 will need to be filed.

e) Once you have filled in the form you must remember to sign and date it.

f) For details of the fee and ways to pay please contact the Patent Office.

PATHFINDING SYSTEM

This invention relates to a method and system for determining a path along a plurality of points in a representation of a virtual world. In particular, but not exclusively the invention relates to a method and system for generating a new path to a destination node in a virtual environment which uses stored information associated with a previous path to the destination.

The new path may be generated to enable a participant in the virtual environment comprising to toggle from a manual navigation mode to an automatic navigational mode. The ability for a participant in a virtual world to switch, or toggle, between automated navigational modes of a navigation system and semi-autonomous or directly controlled modes of the navigational system is very desirable. However, it is also desirable for a user to be able to toggle between navigational modes of navigation systems arranged to enable a user to navigate along a path in the real world, as well as systems which enable a user to navigate along a path in a virtual environment.

It is known in many prior art real world system, such as, for example, car navigation systems, for a route to be determined by a user entering a start point and a destination point, and for the navigation system to dictate directions for the user to follow which constrain the user to following a predetermined path from the start to the destination. If, however, the user wishes to leave the predetermined path, for example, to perform a detour, the navigational system will attempt to redirect the user back towards the destination. Usually, the navigational system will seek out the closest point along the original route for the user to return to, and the user will then resume their original path.

Systems in which 3-dimensional virtual reality environments (virtual worlds) are defined and created are becoming increasingly prevalent in the current technological age. Such computer systems are applicable in numerous commercial applications, such as for online tourist sites, fantasy worlds, gaming, architectural walk-throughs, estate agency (virtual tours of houses for sale), and many others. Typically, a virtual world might be defined by multiple x,y,z, co-ordinate sets which together map out an environment in three dimensions (x, y, and z axes). Objects and walls (or other items which exist within the virtual world), which a user navigating through the virtual world is not allowed to walk through, can be implemented by what is termed a collisionable mesh. The collisionable mesh may therefore comprise a further set of three dimensional co-ordinates defining shapes within the virtual world which the user must navigate around.

Navigation is required within virtual worlds for any entity passing through that world, such as a computer generated person, vehicle or gaming character. This is typically under the control of

an external human user. Traditionally, many commercial implementations of virtual environments have been built on the basis of the self-navigation techniques inherited from games engines, where exploration of all areas of an unknown virtual world is expected, and also a high degree of user competency is assumed. However, as the application of virtual worlds spreads into other commercial areas, it is important that the functionality of navigation evolves to suit the competency and requirements of a whole new wider group of users. Such users may not be as proficient in the navigation around virtual worlds, and furthermore may be disinclined to spend large amounts of time familiarising themselves with the environment. They are likely to become disorientated or bored much more quickly, or even miss important content entirely, unless the user experience is significantly improved.

In our day-to-day lives, humans typically employ various techniques to help us with our navigation when moving from place to place. We retain an idea of our location and orientation using fixed points of reference (landmarks), measures of distance travelled and changes in direction so as to prevent ourselves becoming disorientated. Such techniques may not be available to the novel user in a virtual world which they are unfamiliar, particularly if teleporting (e.g. instantaneously transferring between two different points in the world) is applied, as is the case in many gaming environments. Consequently, it is advantageous to provide an automated pathfinding system for the inexperienced user to help them navigate them through the virtual world, allowing them to overcome the problems of lack of familiarity and any frustrations with the traditional interface of self-navigation. Consequently, many systems have been developed which automatically calculate the best path through a virtual world, for example to find the shortest path to view a particular object or to pass through a room.

PRIOR ART

Barber describes a navigational system in United States Patent Application No. Us 2002/0175918A1, entitled "Method as System for implementing a path network in a computer graphics scene". Barber et al describe a method for implementing a path network for controlling motion behaviours of at least one object in a computer graphics scene. The path network is formed by a plurality of nodes and segments. Various parameters affiliated with the nodes or the segments are defined for establishing motion conditions for the object to travel along the path network. Common to all modes of the path network that Barber et al describe are the parameters EntryPoint and ExitPoint. An Entry Point is a specified path network node or other predefined position on the path network. However, Barber et al do not disclose any specific algorithms to direct an object to get on or off the path network, and the entry and exit points that Barber et al describes are predetermined by the designer of the system. Thus it is not possible in a world such as Barber describes for a user to follow a guided path along which they are automatically navigated, and then to toggle into manual mode and stray from the path, and then toggle back to automatic mode and resume their original path.

Poppen et al describe in United States Patent No. 6,038,509 a pathfinding system which provides a user with a set of directions for following a path from an origin to a destination in a network. If the user deviates from the path, the system recalculates a new path which directs the user from the user's new location, which is off the original path, to the destination. The system adds links to the network to decrease the amount of time needed to determine the new path to the destination. A new path is determined from the new location of the user to the destination using the augmented network. In order for the system to calculate a new path, a new origin is determined either at a pre-existing node where the user is located, or at a pre-existing node of the network which the user specifies as the new origin, or the system can determine the new origin to be some distance away from the user's current position so that the user is able to be at the estimated new origin at when the method for recalculating the path is completed. However, the network which Poppen et al describe comprises a nodal system in which the nodes are fixed and predetermined. The pathfinding system is constrained by the fixed topology of the network which constrains the path to specific nodes of the network.

Known pathfinding algorithms work by expanding possible paths, one node at a time until a path from the start node to the destination node is found. The order in which the nodes are attempted is critical to the speed and quality (e.g. length) of the final path that is found. In a breadth-first techniques, the algorithm searches firstly through all a nodes' immediate neighbours, and then moving onto the neighbours of those neighbouring nodes. In contrast, in depth-first techniques, the algorithm searches for a path recursively using a child node each time until reaching a predetermined depth, at which point it retraces it's steps and tries other child nodes. A best-first search chooses paths preferentially on the basis of an estimate of the shortest remaining distance to the goal.

The article entitled "Middleware Solutions for Artificial Intelligence in Computer Games" (Chapter 3 – AI Techniques) by Jan-Harald Fredriksen of the Norwegian University of Science and Technology, located at http://www.idi.ntnu.no/grupper/su/fordypningsprosjekt-2003/fordypning2003-Jan-Harald-Fredriksen.pdf is concerned with pathfinding techniques in virtual worlds, and the use of the A* algorithm. This document considers the use of cost constraints for different terrains to be taken into account when calculating a path through the virtual world, and the use of hierarchical pathfinding (finding a broad overall route before computing the detailed portions) and portals (which split regions of the virtual world into smaller parts) for improved computation performance. The underlying structure of the virtual world on which the pathfinding in this document is based has two alternative versions. The first is referred to as the POV (points of visibility) approach, in which a plurality of nodes (called waypoints) are introduced into the virtual world, and which the path is found on the basis of line-of-sight links between the nodes. The second approach is the use of a navigation mesh in which

a plurality of convex polygons cover the surface of the world, and in this case the pathfinding algorithm considers each polygon to be a node and calculates a path through the virtual world accordingly.

In addition, there currently exist systems for automating the plotting of travel routes through parts of the real world. A good example of such a system is the "Route Planner" tool provided on the web-site of The AA organisation (see http://www.theAA.com). Such systems take into account certain preferences of the user (e.g. to avoid motorways, shortest distance, shortest time, etc). However the present inventors are not aware of any systems which enable routes to take account of less utilitarian preferences of users in anything less than a very rudimentary manner. Also, such systems are necessarily restricted to calculating a route along predefined paths (i.e. where roads exist), and are therefore restricted in the flexibility of the final route that is generated.

The invention seeks to obviate and/or mitigate the limitations of known navigational systems which provide means to toggle between automated and manual navigation modes. The navigational toggle system according to the invention is implemented in a system comprising a dynamically varying topology of nodal points, where a guided path may be generated dynamically. As the position of the nodes in the environment and the network of potential links between nodes of the environment is not predetermined, the user is provided with more freedom to toggle between automated and manual navigation modes.

SUMMARY OF THE INVENTION

The aspects of the invention are as described by the accompanying independent claims whose dependent claims represent preferred features of the invention.

Nonetheless those skilled in the art will appreciate that the preferred features may be suitably combined with any of the aspects of the invention in any appropriate manner apparent to those skilled in the art.

In the navigation system according to the invention, the position of navigational points of the network is not predetermined. Moreover, the navigational paths are which are generated within the topology are not confined to a predetermined configuration. Thus when a user toggles from an automated navigational mode to move away from a predetermined guided path the topology, the nodal points of the topology may be redefined according to the user's change of position. Moreover, when the user wishes to resume an automated path and toggles back from a manual navigation mode to a navigational mode in which the user is automatically navigated along a guided path, the navigational nodes which exists within the network at the point when they left

the first guided path may have been redefined.

In addition, the user is not constrained to exit the guided path at one of the navigational nodes of the network. Instead, the user is able to switch freely at any point to a manual navigation node. An exit node may be created at the point where a user leaves the original guided path. The exit node may be a navigational type of node augmented with one or more specific features, such as for example, a specific type of sphere of influence. More detail on the effect of sphere's of influence within a pathfinding system is described in the inventor's copending patent application entitled "PATHFINDING SYSTEM", a copy of which is filed herewith as Appendix I as part of this specification.

For example, in one embodiment of the invention, a sphere of influence node is dynamically generated around an exit node created when the user toggles to a manual navigation mode. If the user navigates themselves beyond this sphere, the option for the user to rejoin the original path generated from the start to the destination may expire. Alternatively, a time-limit on the time the user spends away from the original path may also cause this option to expire. In either case, the option to rejoin is automatically disabled as the user will have wandered too far away (or has spent so long away), that the original path is now obsolete. For example, if a user has wandered to the other side of the original destination, so that the user would want to approach the destination from a completely different angle, it would not be an efficient use of computational time/performance for a search to be performed to determine an optimal path rejoining the original path.

The navigation system is provided within a virtual environment within which the density of points within the virtual environment which define a path between an initial starting point and an initial destination are not fixed and which can vary according to a number of conditions, such that the topology of points determining a path between two points can be dynamically changed as the path itself is plotted. This enables a path to be provided within a virtual world environment along which a user can be automatically navigated. The topology of the path and/or the speed at which the user is automatically navigated along the path enables the user to encounter one or more objects of interest.

One or more objects which the guided path seeks to guide the user towards may be dynamically varying (for example, in terms of their position within the virtual world). This is particularly so in virtual environments where a user may be able to interact via characters within the virtual world with each other. For example, if one of the characters is controlled by another user participating within the same virtual environment, the dynamic path may change if that character is retired from the virtual world by the user controlling it.

This invention is applicable to any type of virtual world, which might include for example fantasy worlds or worlds based upon a real environment such as a historic building or design for a future building, or a house that is for sale , and in which the obstructions might comprise walls or other obstacles. The representation in the sense used above is likely to refer to stored digital data that represents such a world.

Advantageously, the arrangements of the invention employ a technique which allows a content provider to define some initial points for navigation in a virtual world, allowing the content provider to influence the routes taken and the items of interest that can be visited by a user. The system then automatically multiplies the number of points in a manner which will improve the user's experience along the subsequently generated path through the environment compared with only a limited set of originally defined navigation points.

In comparison with previously known systems in which a path through a virtual world is not constrained to specific navigation points, the embodiments of the invention provide improved processing during the actual pathfinding procedure. This is because no collision detection (ensuring the path avoids obstacles, etc) is required during pathfinding in the embodiments since all the new navigable nodes and links have been added on line of sight principles, and therefore by definition already avoid all the potential obstacles.

Additionally, these embodiments provide advantages over previous known systems such as the AA Routefinder which are necessarily limited to finding routes along specified predefined roads, etc. In contrast, in the embodiments, the user is able to navigate within all areas of the virtual world.

BRIEF DESCRIPTION OF THE DRAWINGS

For a better understanding of the present invention, specific embodiments will now be described, by way of example, with reference to the accompanying drawings, in which:

Figure 1 shows a schematic of a system for pathfinding in a virtual world environment;

Figure 2 shows a diagrammatic representation of a virtual world environment;

Figures 3 to 12 show further diagrammatic representations of the virtual world environment of Figure 2; and

Figure 13 shows a flow chart of the method for processing points within a virtual world;

Figure 14 shows a flow chart describing a method of toggling between navigational modes within a virtual environment according to a first embodiment of the invention;

Figure 15 shows a flow chart describing a method of toggling between navigational modes within a virtual environment according to a second embodiment of the invention;

Figure 16 shows a flow chart describing a method of toggling between navigational modes within a virtual environment according to a third embodiment of the invention; and

Figure 17 shows a flow chart describing a method of toggling between navigational modes within a virtual environment according to a fourth embodiment of the invention.

Figure 18 shows schematically a virtual environment in which a user has left an automated path according to one embodiment of the invention;

Figures 19A, 19B and 19C show schematically how a return path to an automated path may be determined according to the embodiment of the invention shown in Figure 14;

Figures 20A and 2b show schematically how a return path to an automated path may be determined according to the embodiment of the invention shown in Figure 15;

Figure 21 shows schematically how a return path to an automated path may be determined according to the embodiment of the invention shown in Figure 16; and

Figure 22 shows schematically how a return path to an automated path may be determined according to the embodiment of the invention shown in Figure 17.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

The best mode of the invention currently contemplated by the inventors will now be described with reference to the accompanying drawings.

The invention enables a path to be generated which includes nodal information derived from a first, previous path. One navigational system which enables the first path to be generated will now be described with reference to Figures 1 to 13 of the accompanying drawings.

Figure 1 illustrates a system 1 for creating, and allowing a user to navigate within, a virtual world. The system comprises a server computer 2, accessible to a content provider (a human user who will input the specification for the virtual world) via terminal 3, and accessible to a user (desiring to view and navigate through the virtual world) via another terminal 4. In this particular embodiment, access from both terminals to the server 2 is through a general network 5, such as for example the internet, although it is understood that the illustrated computer arrangement may be replaced by any suitable alternative which allows a content provider to design and implement a virtual world, which a user can subsequently access and interact with.

A schematic of a simple virtual world environment is shown in Figure 2. This represents a room 20, including walls 21, doors 22, and various obstacles 23, 24 and 25 within the room. When specifying the design of this virtual world 20, the content provider inputs data in the form of co-ordinates representing the shape of the environment into the server 2 via terminal 3. The data might include co-ordinates representing a collisionable mesh defining the obstacles 23, 24, and 25 through which a navigating user is not allowed to pass.

Feature Nodes

The content provider is able to define features of interest within the room that a user may wish to view. For example, if the virtual world represents a tour through a historic building, it may be desirable to insert features such as statues, furniture, paintings and information boards. These may be represented by objects within the room, for example statue object 26, painting 27, and information boards (obstacle 25).

In order that these features may be visited by a user, the content provider defines feature nodes 28A, 28B and 28C (shown in Figure 3) associated with the features, which the user can view or interact with. A feature node is therefore essentially a point in the virtual world, linked either to an object or to the collisionable mesh, and in the embodiment has the following data set:

Node ID
Feature Group ID
Object ID of the linked mesh / object
XYZ of object pivot point data
Sphere of influence settings
Display settings

Node ID – an unique identifier for the node (e.g. text string "monet_painting_01" or numerical identifier "00563")

Feature Group ID – an identifier for the type of feature associated with the node, for example all paintings may be in a first Feature Group identified as "paintings", whilst information boards might be a second Feature Group "information_points". The choice of groupings is highly dependent on the content provider and the type of features they wish to distinguish within the virtual world.

Object ID of linked mesh / object – this is the identifier for the object or collisionable mesh associated with the feature node, and may be the same as the node ID (e.g. "monet_painting_01")

XYZ of Object Pivot Point Data – these are the co-ordinates giving the location of the feature node (and may be obtained by extraction from properties of the object / mesh with which the feature node is associated)

Sphere of influence settings – these define (for a 3D world) a three dimensional sphere (for example, illustrated as circle 31 in Figure 3) surrounding the feature node, and are used during the calculation of navigation paths through the virtual world

Display settings – these are some of the settings to be used for text / images associated with the feature

Navigational Nodes

The content provider also initially defines a plurality of navigational nodes (29A, 29B, 29C, etc) within the virtual world, as shown in Figure 3. Essentially, these provide a simple matrix of navigational points which the automated path generation algorithm can use to plot a route through the environment. Associated with each navigational node is the following data set:

Node ID
Node Level ID (optional)
XYZ position
LoS data
LoS constraint settings (optional)
FoV constraint settings (optional)
Orientation constraint settings (optional)

Node ID – a unique identifier for the node

Node Level ID – this is an identifier which allows a level for the node to be defined, effectively grouping the nodes into different subsets, and used during the path finding process to split the virtual world into different regions for processing efficiency. For example, two different rooms in the virtual world could be represented by different regions of nodes, with the doorway between them comprising a switch node. For processing efficiency, a path can initially be calculated based on only the nodes of one of the rooms (thus saving erroneously calculating at this stage paths which enter the second room, and only later allowing the pathfinding calculation to extend to the second room via the switch node). Additionally it is possible that the different sub-sets could cover overlapping regions (for example a first level of nodes might provide coarse coverage of a large area while different levels might provide finer coverage of the same area or subsets of that area).

XYZ position– these are the x, y, z co-ordinates giving the location of the navigational node

LoS data – this data field will be empty initially, but is later used to contain a list of all nodes which the present node has Line of Sight (LoS) to, i.e. all the nodes that it is able to link to (see later for further details)

LoS constraint settings – the content producer may choose to manually define an angular LoS constraint (e.g. an angle range, in which the node is only allowed to search for nodes to link to if they fall within that viewing angle, see later for more details)

FoV constraint settings – the content producer may choose to define angular constraints for the Field of View (FoV) which will constrain the range of images allowed to be displayed to a user at that node (e.g. for node 29F, the FoV may be constrained in the angular direction towards statue object 28A to ensure that the user sees the object)

Orientation constraint settings – for some navigational nodes, the content producer may choose to constrain the allowed orientation of the user as they approach the node, by defining orientation constraint settings this forces the calculated path to approach the node from a particular direction.

The content provider is thus able, through the use of navigational nodes, to define key positions within the virtual world from which the potential user is able to best experience the features of interest within the environment. The content provider can define the exact position, direction of approach (orientation settings), and viewing angle (FoV constraint settings) so as to increase the quality of the user's experience of the features of interest, compared with other systems which are completely automated.

Processing

The subsequent processing is explained with reference to the flow diagram in Figure 13. After node data has been input to the system (step 40), a node matrix is automatically created for the virtual world (step 41). This identifies for each navigational node all the other navigational nodes it is able to link to. Node links are calculated following basic rules of LoS (line of sight) for two nodes. This takes into account the collisionable mesh and any other objects deemed to crop the LoS of the nodes (i.e. no links are allowed to pass through obstructions because a path between nodes cannot pass through these). Figure 4 illustrates all possible LoS links for navigational nodes in the virtual world 20 (i.e. the lines linking nodes 29A, 29B, etc). For example, Node 29A has direct line of sight to both nodes 29B and 29F, but not to 29C because this would pass through the obstacle 23. A node matrix for virtual world 20 of Figure 4 is represented in tabular form below:

Node	Links to other nodes
29A	29B, 29F
29B	29A, 29C, 29D, 29E
29C	29B, 29D, 29F
29D	29B, 29C, 29E
29E	29B, 29D, 29F, 29G
29F	29A, 29C, 29E
29G	29E

At this stage each navigational node's data set is provided with LoS data on all its visible partners, including their XYZ position. Any navigational node which does not have line of sight to any other navigational nodes is deemed invalid and may be deleted from the system.

If the content provider had specified any LoS constraints for particular navigational nodes, then this would crop the angle it is allowed to search for node links. This is controlled using "LoS constraint settings" for the node, comprising degree values which slice the otherwise allowed 360 degree LoS. In this case, the node matrix above would differ as some of the links shown there would not be allowed. This functionality allows the content provider to create channels of nodes that ignore other nodes around them, and to produce breaks in the node matrix where nodes are not desired. This can improve the user experience because in this manner the content provider can influence the routes taken by a user within the virtual world.

The next stages of processing will add in additional nodes on the basis of the LoS links already defined. However, before that occurs, the system checks all the links (step 42) to decide which links will / will not be used as a base for the new nodes. In particular, the system looks for any

crossed links (such as the 29B-29E link which crosses the 29C-29D link), and marks the longest of the crossed links as not being available for use as a base for the new nodes. In this embodiment a data flag is associated with the link (in either the node matrix or the individual nodes' LoS link data) to indicate that this link is "not to be subdivided". The affected link is shown as a dotted line in Figure 5.

The next stage of processing involves the automatic creation of a more dense matrix of nodes (step 43). This is achieved by subdividing the LoS links between the existing navigational nodes so as to add new navigational nodes (also referred to as subdivided nodes) at each midpoint (step 44). Figure 6 shows these subdivided nodes (30A, 30B, 30C,... etc). As more and more new nodes are added during each cycle through step 43, the density of nodes increases, up to a maximum allowed node density. This density may be predefined in advance by the content provider, for example by specifying a minimum allowed distance from a node to it's nearest neighbour. After each new node is added, the system checks whether the new node meets the density criteria (step 45). If it falls outside the allowed density criteria, the new node is deleted (step 46). Otherwise, the node is kept. Once every LoS link has been examined for subdivision (step 47), the system checks whether any new nodes were added during the last pass (step 48). If no new nodes were created during a pass, the system terminates (step 49).

For each pass, after the new nodes have been added, the LoS data for every node (both the navigational nodes and each newly created subdivided node) is updated to take account of all the multiple new possible LoS links (step 50). Figure 7 illustrates the newly created dense matrix with many more possible LoS links between the nodes.

The system now checks which of the newly added nodes are valid (step 51). A node is defined as valid if it links any two other nodes which do not otherwise have line of sight to each other. This is illustrated for node 30A in Figure 8. In this drawing, for clarity, the only nodes and links illustrated are node 30A and the seven other nodes it links to. It can clearly be seen that node 30A acts as a bridge for many different pairs of nodes (i.e. it forms a connection between two nodes which do not have LoS to each other), such as between nodes 29B and 30F. Therefore node 30A is valid, and in fact only requires that it is connected to one pair of nodes which do not have LoS, to be defined as a valid node. Any invalid nodes are deleted.

Figure 9 shows the same procedure for node 30B. Node 30B is defined as valid. However at this stage, any of 30Bs links which do not assist in bridging between two nodes are deleted. This is the case for the link between node 30B and 29C. Since node 29C already has line of sight links to all the nodes shown on Figure 9 (and reciprocally node 30B also has line of sight to all the nodes linked to 29C), therefore link 30B-29C cannot help to bridge between any pair of nodes that do not have line of sight to each other, and the link 30B-29C is deleted.

Link 30B-30C would also appear from Figure 9 to be redundant, because node 30C already has line of sight to all the nodes in Figure 9. However, link 30B-30C is actually valid because it bridges, for example, nodes 30B and 30A (which do not have line of sight to each other) which can be seen from Figure 7, and therefore link 30B-30C would not be deleted.

Validity checks are carried out in this manner for all new nodes, and two others (30E and 30H) are shown, in Figures 10 and 11 respectively.

After the validity checks have been performed on the new nodes, the system again checks for any crossed links (step 52). These are the links which are not to be used in any subsequent cycles of subdivision. The links are identified as before, by looking for links which cross each other and flagging the longest ones as "not to be subdivided". This has been performed for all the links, and the resulting link matrix is illustrated in Figure 12 (where dotted lines illustrate those links which are not to be used in the future for subdivisions to add more nodes).

Thus one cycle of subdivision has been completed, and even after this, the number of nodes has more than doubled, and the connectivity between the nodes has greatly increased. For example node 29D previously only had links to 29B, 29C and 29E, but is now additionally linked to 30B, 30C, 30E and 30F.

Further cycles of subdivision (step 43) may be repeated, until the system has created a suitably dense matrix of nodes which map out the non-collisionable environment in the virtual world. Advantageously, during the subsequent pathfinding calculation, there is no need to calculate collision data in real time because the matrix nodes are all constrained to navigable areas. In addition, with the higher density of nodes in the environment, the system can generate far more efficient paths than with the originally defined node matrix, and which enhance the user experience by giving a much smoother path through the virtual world which typically follows the type of behaviour a user is used to when navigating a real world environment.

Once the path has been calculated a spline is generated to further interpolate and hence to smooth the journey still further. In the present embodiment, once the spline has been generated along a particular path, a default speed along the path is set. In the present embodiment this is set to vary between a fairly fast speed in areas of little interest and to a relatively lower speed around features of interest. Ideally this is done by smoothly varying the speed to have minimum values at the feature nodes of most interest and to have maximum speeds in between these points.

Although in the embodiment above, the node density criteria is specified in terms of the minimum allowed distance between a new node and an already existing node, this could be

replaced by any other suitable mechanism for specifying and testing the node density. Another possible method is to specify in advance a maximum allowed number of cycles of subdivision before the algorithm must terminate.

Although In the embodiment above, the check for whether a new node is valid involves checking whether the node acts as a bridge between two other nodes, an alternative/additional criteria which may be used to define whether each newly added node is valid is to check whether the new node assists in providing a shorter path between any pair of nodes.

Pathfinding

As mentioned earlier, the A* algorithm is an optimal pathfinding algorithm both in terms of the processing time required, and the quality of the path found. Start and destination nodes are defined in advance, and the algorithm explores multiple different paths to find the best route from the start to the destination. Consider the case of a user who wishes to tour through virtual world 20, entering at door 22, and who has indicated via a suitable interface that he desires to approach information board 25.

The start node is defined as node 29A, and the destination node as 29G. The A* algorithm operates by maintaining two lists of nodes: a list of nodes that have already been explored (the Closed list) and a list of nodes linked to the ones that have been explored but have yet to be explored themselves (the Open list). The algorithm takes a node from the Open list, and if it is the destination node then the algorithm terminates. If not, all the nodes linked to the chosen node are added to the Open list. The A* algorithm is known as a heuristic method because it uses estimates to guide the search for nodes. In particular, in order to decide which node should be selected next from the Open list, all the nodes have an evaluation function f = cost + heuristic, and the node which is selected as the next current node from the Open list is the one with the lowest value for f.

The cost is a measure of the quality of the path so far from the start node to the current node (i.e. for a simple A* algorithm this could be the sum of the lengths of all the links in the path between the start node and the node). The heuristic function is an estimate of the cost to get from the current node to the destination node. The results of the A* algorithm are very much dependent on the choice of cost and heuristic functions, and it is the specific weighting and cost values used in the evaluation function for the embodiment of the invention that allow the optimum path to be generated.

The cost function of the embodiment takes into account various factors of the route, together with any user specific preferences. The cost function is a weighted sum of the following form:

cost = w1 * distance + interest_component + w3 * angle

where distance is length of the links so far, and w1 is the weighting ascribed to the current user indicating how important it is to them that the path is as short as possible. Interest_component gives a measure of how interesting the node (s) so far have been (but since we are looking to minimise the cost function, the lower the interest_component value the more interesting the nodes so far have been). Thus, in the present embodiment a summation over the node or nodes included in the partial path built thus far of the form:

$$\sum_{PARTIAL\_PATH}(1 - w2 * Interest)$$

is used, in which w2 is the percentage weighting interest given to the feature group associated with the node in question, and Interest is a measure of how strongly a node is associated with the corresponding feature associated with the node. Thus, in the present embodiment, feature nodes (typically of great interest to a user) may be given an Interest value 1, whilst navigational nodes have interest value of 0 and therefore when summed as above give a higher cost. As mentioned, weighting w2 gives a measure of how important it is to the user that they visit features of interest of each respective feature group.

Finally, angle gives a measure of how much the path so far wiggles, since this represents a sum (or average) of how much the angle between the entry and exit links for each node deviates from a straight line. Weighting w3 gives a measure of how important it is for the user that the path wiggles as little as possible (i.e. a zero value of w3 would suggest that the user did not mind how wiggledy the path was whilst a high value (eg 100%) would indicate that the user would prefer to avoid wiggly paths).

The heuristic is an estimate of the cost to get from the current node to the destination node, and one possible choice for this is:

heuristic = euclidian_distance_to_destination

The heuristic in the embodiment is therefore based on the Euclidian distance to destination (i.e. a straight line from the current node to the destination node), although it could alternatively also include some measure of the distance, interest and angle weightings w1, w2 and w3 if desired.

The system therefore runs through the A* algorithm for the nodes using the evaluation function, until the final destination node is reached. Importantly, the weightings w1, w2, and w3 can be

varied depending on the user profile.

Defining a user profile

A suitable interface (not shown) allows the user to define their profile (e.g. what they are particularly interested in, and how they like to navigate through the virtual world). The user profile settings are then incorporated into the weightings above used for the A* algorithm, which consequently will result in different paths being generated for different users. The translation of the user interests into weightings allow different routes to be compared against each other such that the optimum path for that user can be generated. It can be seen that for a user who is particularly interested in visiting features of interest (paintings, landmarks, etc) it is likely to be the case that the system will not calculate the quickest or shortest route between two points. Rather, the lowest cost path (i.e. the best route for that user) will be one which visits as many features of interest as possible.

The user may input their profile upon first entering the virtual world. Some options include:

Always select shortest path	on / off
Always select quickest path	on / off
Ignore all features	on / off
Gaze orientation	on / off
Velocity control	on / off
Importance of path being short (w1)	weighting
Importance of path not being wiggly (w3)	weighting
Importance of visiting Feature Group 1 [e.g. paintings] (w2)	weighting
Importance of visiting Feature Group 2 [e.g. information points] (w2)	weighting
Importance of visiting Feature Group 3 [e.g. landmarks] (w2)	weighting

The first five options are overrides, and are thus non-weighted. For example, if the user selected "Always select shortest path" as ON, then the weighting w2 for the features of interest become irrelevant and will be set to zero.

The final five options are all weighted, and however the weightings are distributed they should all be normalised (e.g. add up to 100% in total). Any option which is not important to the user and not given a weighting is deemed irrelevant and not taken into account in the pathfinding process.

Dynamic Navigational Nodes

In addition to the fixed navigational nodes (29A, 29B, 29C, etc), defined initially by the content provider, the system may also include dynamic navigational nodes (not shown). These nodes operate like navigational nodes in the sense that they can form part of the path generated for the user, but provide enhanced functionality because the user is able to move them. A dynamic navigational node is associated with an interactive mesh object in the virtual world environment that can be moved around by the user. The user is able to specify that they want the dynamic navigational node to be a destination point, and the path will be calculated to it's location.

If a dynamic navigational node is placed a large distance from other navigational nodes, then the subdivision process described earlier may be used to fill in new nodes until a suitable node density is achieved in that area. This then allows a smooth path to be calculated to the node.

If a dynamic navigational node is placed without a line of sight to any other navigational nodes (i.e. in a blind spot) then the node is defined as invalid – lost from the Node Matrix (although still present in the environment) and the user is prevented from selecting it as a destination. Alternatively, the system may prevent the user from placing the node in a blind spot, and only allows it to be moved into valid positions.

Switch Navigational Nodes

In addition to the fixed navigational nodes and the dynamic navigational nodes, the system may also include switch navigational nodes (not shown), defined by the content producer. As mentioned earlier, navigational nodes have a Level ID, specifying which level the node belongs to, and effectively grouping the nodes into different subsets for more efficient path calculation. For example, if the virtual world consists of two different rooms, then the navigational nodes in one room may be defined as belonging to one level, and the navigational nodes in the second room to a different level. During the pathfinding process in one room, only the nodes of that level are evaluated, saving takes into account nodes on a different level. The switch nodes then provide the functionality to swap from one level to the second level (e.g. the switch navigational node might be positioned in a doorway between the two rooms).

Associated with each switch navigational node is the following data set:

Node ID
Node Level ID #1
Node Level ID #2

XYZ position
LoS data
Switching data

Node ID – an identifier for the node

Node Level ID #1 – the identifier for the first level

Node Level ID #2 – the identifier for the second level

XYZ position– these are the x, y, z co-ordinates giving the location of the switch navigational node

LoS data – this gives the data for all nodes (for both levels) which the present node has Line of Sight (LoS) to

Switching data – gives the criteria of when the generated path can / cannot switch from navigational nodes on one level to nodes on the other level

Event Feature Nodes

As described earlier, the content provider is able to define feature nodes (28A, 28B, 28C, etc) associated with the features of interest in a virtual world. The content producer is also able to define event feature nodes (not shown) which have the extra functionality that they react to users within the virtual world, or are only active at certain predefined times.

These event feature nodes expand the functionality of the system into the gaming and interactive online environment platforms, where multiple users can interact with the environment and with each other (e.g. the Event Feature may simulate an explosion, or the opening and closing of an interactive chat-room / race event. The content provider must specify when the event feature will be active / inactive, or what triggers it will respond to. An example data set is as follows:

Node ID
Feature Group ID
Trigger data
XYZ of pivot point data
Sphere of influence settings
Time settings
Display settings

Node ID – an identifier for the node (e.g. text string "explosion")

Feature Group ID – an identifier for the type of feature associated with the node

Trigger data – the conditions which will cause the event feature node to become active

XYZ of Pivot Point Data – these are the co-ordinates giving the location of the event feature node

Sphere of influence settings – these define (for a 3D world) a three dimensional sphere surrounding the feature node. However, in addition to the settings used for the standard feature nodes, these additionally include a time dependant feature which may allow the sphere of influence to degrade over time.

Time settings – these define when the event feature node will be active / inactive

Display settings – these are the settings to be used for text / images associated with the feature when constrained

Further functionality

As explained above, the processing and pathfinding stages result in a spline path which will be used to take the user through the environment. The spline path therefore comprises a plurality of nodes each joined by links to the next node. The system applies smoothing to this path by interpolating the spline so as to improve further the user experience.

During real time display, when the user is guided through the environment they have the further option to control their field of view. As discussed earlier, the content provider can specify particular angles of viewing (FoV constraints) which are to be applied for specific feature nodes so as to ensure the user does not miss features of interest. The user can choose to toggle control to either operate this automatic field of view presentation or alternatively to control manually their own field of view.

In the embodiments described above, it is explained that the processing stage which creates the more dense matrix of nodes occurs after the content provider has defined the contents within a virtual world, but before the system is used by a user desiring to navigate within that world. It is of course alternatively possible for either part or all of the processing stage which adds these subdivided nodes to be carried out in real time whilst the system is in use by a user. Indeed, an alternative arrangement is particularly desirable when the virtual world contains a number of dynamically moving objects (for example, a virtual environment of a city might include moving cars, buses, etc). In this dynamic case, only some subdivision (for the static objects such as buildings, etc) could be carried out in advance before a user accesses the system, with

the remaining processing to add the extra nodes occurring each time a user specified that they desire a path to be created. The processing stage thus takes into account the positions of these dynamic objects at the instant the user desires the path to be created. This functionality could be supplemented by defining some regions in the world in which this subdivision will occur (for example, streets within a virtual city), and other regions in the world when the dynamic subdivision will not occur because they contain only stationary objects (for example, the insides of some buildings). This is a typical example when switch nodes (mentioned above) are used to delimit the different regions.

Furthermore, it may be necessary in real time to create new nodes corresponding to the start and destination points selected by a user for a path generation procedure. If nodes do not already exist at these locations then the system may be used to automatically create such nodes, and additionally use processing to add new subdivided nodes in only the regions close to the start and destination nodes so as to link them to the already existing node matrix.

The method of generating a new path to a destination node in a virtual environment comprising a plurality of nodes will now be described in which nodal information identifying one or more nodes associated with a previously created path to said destination node is processed to determine the new path. The processing may comprise simply updating the spheres of influence and/or the interest_component of one or more of the nodes associated the previous path and repeating the pathfinding process described hereinabove. Alternatively, more complex processing may be done to on said stored nodal information to determine the new path.

In particular, the new path may enable a user who was being navigated automatically along the previous path and who switched from an automatic navigation mode to a manual navigation mode to resume an automatic navigation mode to the destination of the previous path. This can be implemented using the above path finding system even where one or more nodes in the virtual environment are changing dynamically, including even if the destination node has moved. In one embodiment of the invention, a validation of the previous path is performed as this may no longer be valid in terms of LoS or other constraints due to changes in the position and/or characteristics of the nodes which were included on the previous path to the destination. This aspect of the invention to reuse previously stored nodal information will now be described in more detail with reference to Figures 14 to 22 of the accompanying drawings.

One embodiment of the invention comprises a method of generating a return path to a portion of a first path created from a start node to a destination node in a virtual environment. The first path can be created by any suitable path generation process in which the nodal topology of the virtual environment can be reconfigured dynamically to include additional navigational nodes at the point where a path is to start.

In Figures 14 to 17, steps are shown associated with a method of toggling between navigational modes (defined in this context to be equivalent to switching or changing navigational modes) within a virtual environment is shown according to a various embodiments of the invention.

The pathfinding system described herein above is used to generate a path from an initial origin to an initial destination (step 60) along which a user is navigated automatically (step 62). Whilst origin of the path may be co-located with a navigational node of the path-finding system, the path-finding system may automatically create a start navigational node at the point where the user requests to be automatically navigated to a destination. Similarly, whilst the destination may be located at a destination navigational node for the automatic path generation, if not, the pathfinding system will generate a destination navigational node where the user has defined the desired destination.

At a point along the path, the path becomes no longer actively navigated along. For example, a user activates a toggle to switch from the initial automatically navigated navigation mode to a manually navigated mode (step 64) in Figure 14. This exit point may or may not be co-located with a navigational node of the path. In the embodiments shown in Figures 14 to 22, a new exit node F is created at the exit point as this lies between the nodes of the path. However, the exit point need not always coincide with a node of the path. In such circumstances, an exit node may be created, or alternatively, the node before or after the exit node on the path may be defined to be an exit node . Figure 18 shows how a user navigating along a path from A to E may toggle off at an exit point between nodes B and C of the path.

Information relating to the nodes of the path is stored when the path becomes no longer actively navigated along. Thus the start node and destination node of the path are saved, together with information relating to any intermediate nodes. If a new exit node is defined/created, information regarding this is stored too. The user is then able to deviate at will from the path and explore using a manual navigation mode until they determine they want to resume an automatic navigation mode (step 66).

In the embodiment of the invention shown in Figures 14 to 17, the pathfinding system will seek to generate a new path to the destination node of the previous path which processes the stored nodal information of the original path when determining the new path. In Figures 14, 15, and 17, a return path to the guided path is determined according to one or more criteria to reduce the computational complexity associated with generating the new path to the destination.

In Figure 16, the nodal information stored is updated with an increased interest_component and a new pathfinding process is performed with this increased interest_component. This makes it

more likely that the new path will intersect the original path. If the new path intersects the previously generated path, it is possible to reuse the stored nodal information to continue on to the destination. Alternatively, as Figure 17 shows, it is possible for the new path to be completely independently regenerated to the destination, but by increasing the interest_component of the nodes associated with the previous path, it is more likely for the user to resume a path similar to the previously generated path. By reusing the nodal information associated with the previous path, the new path to the destination can be determined more quickly and efficiently, preventing the user from experiencing an undesirable delay before resuming the automated navigational mode. This is particularly advantageous when a path is being generated dynamically.

According to the invention, the path generated should be the optimal path permitted by the pathfinding system. This may be the shorted path back to the original guided path, but may not always be due to the possibility for the topology of the network of nodes defining the automated path system being capable of dynamically changing as a user progresses along a path.

Due to the potential for the network topology itself to evolve autonomously with the virtual environment, the original path is revalidated at the point where the user selects to toggle back to the automated navigation mode (step 68) to determine if the features of the original path are still relevant for the original path to match the user profile options.

If the original path is still valid, then a new start node is created at the toggle back point (the point where the user wishes to resume automated navigation) (step 70). However, if the path is invalid, then the original path data is discarded (step 72). In this latter case, a fresh path search retaining only the destination node is initiated (step 74). Alternatively, the user may be able to continue to rejoin the original path but a warning is displayed to indicate this is now no longer recommended (step 72).

If original path is still valid, the new start node created at the toggle back point, which is then attached to the network to be used to generate a return path using the pathfinding system described hereinabove.

In Figure 14, the return path is generated by performing a parallel node search from the new start node to all nodes which are defined along the original path between the exit node to the destination node of the original path. These nodes need not be in the same position along the path when it was originally created, i.e., the actual topology of the original path may have evolved. It is also possible that certain new nodes may now exist along the originally guided path and/or for some of the nodes along the originally guided path to no longer exist. Thus if a node associated with a moving object having a high interest feature weighting or a large sphere

of influence has moved, the original path may have changed accordingly. In one embodiment of the invention, once the nearest node along the original path has been determined, no other nodes along the original path are determined (step 78). This enables all nodes prior to the nearest node to be discarded, and only subsequent nodes which remain along the guided path to the destination remain.

A return path is then generated using the pathfinding system to the nearest node identified in step 78 and this is combined with the subsequent nodes remaining along the originally guided path to the destination (step 80)

Once the new guided path has been generated in step 80, the navigational system then automatically navigates the user along the new guided path towards the destination node (step 82). When the user reaches the destination node (step 84), the automatic navigation stops and the user resumes manual navigation (step 84).

Figures 15, 16, and 17 retain the same numbering scheme used in Figure 14 for equivalent steps in these different embodiments of the invention.

In Figure 15, another embodiment of a method of toggling between navigational modes within a virtual environment is shown in which step 78 of Figure 14 is replace by step 88.

According to the embodiment shown in Figure 15, if the original path is still valid a new start node is attached to the network of nodes in an equivalent manner to described in step 76 in Figure 14. However, the return path is generated by performing a path search to the exit node only of the original path. The remaining points along the original path data are retrieved, and all points prior to the exit node are discarded. This option may not be available if the exit node has an angle of entry associated with it, and the user has specified only relatively straight paths are to be generated with the virtual environment by imposing an angle differential constraint (for more details see the inventor's copending patent application, entitled " Pathfinding System", a copy of which is filed herewith as Appendix II.

In Figure 16, another embodiment of a method of toggling between navigational modes within a virtual environment is shown in which step 78 of Figure 14 is replace by step 90.
In this embodiment, new interest nodes are created using the Sphere of Influence techniques described in Appendix II at intersecting points to the original path nodes. This results in these nodes having a higher interest weight when conducting a fresh path search (i.e, the interest_component increases). These interest nodes are deleted after one or more toggles have been activated by the user to switch between navigational modes, or if the next path search is targeted at a new destination. However, by increasing the interest_component the

newly generated path is more likely to encounter the nodes of the previously generated path.

In Figure 17, another embodiment of a method of toggling between navigational modes within a virtual environment is shown in which step 78 of Figure 14 is replace by step 92, and a new return path is generated.

In Figure 17, a fresh path search is performed to the original destination which is interrupted as soon as it encounters any node which lay on the original path. This node is then the re-entry node for the original path and all nodes which existed prior to the encountered node along the original path are deleted (step 92).

Figure 18 shows an exemplary path from an origin A at which a start node has been created to destination E at which a destination node has been created. The path generated intersects navigational nodes A,B,C,D and E and circumvents an unnavigable object which prevents a more direct path from being generated from A to E.

In Figure 18, the solid line indicates the guided path which is originally generated from A to E and along which the user was being automatically navigated. The dashed path represents the point along the guided path at which the user switched to manual navigation. Also shown in Figure 18 is exit node F (indicated by the hatched node icon) which was created when the user switched to the manual navigation mode from the automated navigation mode. This is exit node was not part of the original guided path, although it forms part of the path created.

Also shown in Figure 18 by the smiley face icon is the location G at which the user decides to resume automated navigation.

Figures 19A and 19B shows how the parallel path search feature of step 78 in Figure 14 may generate a search for several paths in parallel to nodes of the guided path, including the exit node (Dan/Martin - is this true or does the exit node only come in with step 88 in Figure 15). The closest node to the user is then determined and used to generate a re-entry point to the original guided path. In Figure 19A, the nearest node is in fact the exit node F. However, in Figure 19B, node D has moved and is now located closer to node F, and so node D is selected as the re-entry point. Figure 19C shows how a return path (also referred to herein as a rejoin path) is generated between nodes G and D. The shaded navigational points in Figure 19C are all deleted and the path continues to destination node E along the original guided path. Once this path has been determined, automatic navigation commences along from node G to node E via node D.

Figure 20A shows an implementation of step 90 in Figure 15. In Figure 20A, a path has been

determined back to the exit node F (via navigational node H for example). The exit node F then becomes the point of re-entry to the original guided path and all previous nodes lying along the original guided path are discarded (these comprise nodes A and B in Figure 20B).

Figure 21 shows an implementation of step 92 in Figure 16. In the embodiment shown in Figure 21, nodes C, D, and E, and F have had their interest_component increased to a higher value. When the pathfinding process is performed to determine a new path from start node G to destination node E, the increased interest_component of these nodes results in the new path generated passing through the nodes C and D as well as navigational node L to the destination node E..

Figure 22 shows an implementation of step 92 shown in Figure 17. In Figure 22, a new path is plotted from new start node G to destination E. This takes in navigational nodes J and K, and then navigational node D. As navigational node D is associated with the original guided path from A to E, at navigational node D the new path resumes the original guided path to E. This reduces the computational complexity associated with computing the new path.

Those skilled in the art will appreciate that the invention as described herein with reference to the above description and appendices may be implemented by many features providing an equivalent function to the features described. Those skilled in the art will also realise that the invention is not limited to the embodiments disclosed herein, but extends in particular to include embodiments where it is apparent that a feature described in one embodiment of the invention described herein can be incorporated or adapted for incorporation into another embodiment of the invention or replaced by any feature having an equivalent function, such that the scope of the invention is as defined by the appended claims.

Appendix I

PATHFINDING SYSTEM

TECHNICAL FIELD

The invention relates to a method and system for determining a path along a plurality of points in a representation of a virtual world.

INTRODUCTION

Systems in which 3-dimensional virtual reality environments (virtual worlds) are defined and created are becoming increasingly prevalent in the current technological age. Such computer systems are applicable in numerous commercial applications, such as for online tourist sites, fantasy worlds, gaming, architectural walk-throughs, estate agency (virtual tours of houses for sale), and many others. Typically, a virtual world might be defined by multiple x,y,z, co-ordinate sets which together map out an environment in three dimensions (x, y, and z axes). Objects and walls (or other items which exist within the virtual world), which a user navigating through the virtual world is not allowed to walk through, can be implemented by what is termed a collisionable mesh. The collisionable mesh may therefore comprise a further set of three dimensional co-ordinates defining shapes within the virtual world which the user must navigate around.

Navigation is required within virtual worlds for any entity passing through that world, such as a computer generated person, vehicle or gaming character. This is typically under the control of an external human user. Traditionally, many commercial implementation of virtual environments has been built on the basis of the self-navigation techniques inherited from games engines, where exploration of all areas of an unknown virtual world is expected, and also a high degree of user competency is assumed. However, as the application of virtual worlds spreads into other commercial areas, it is important that the functionality of navigation evolves to suit the competency and requirements of a whole new wider group of users. Such users may not be as proficient in the navigation around virtual worlds, and furthermore may be disinclined to spend large amounts of time familiarising themselves with the environment. They are likely to become disorientated or bored much more quickly, or even miss important content entirely, unless the user experience is significantly improved.

In our day-to-day lives, humans typically employ various techniques to help us with our navigation when moving from place to place. We retain an idea of our location and orientation using fixed points of reference (landmarks), measures of distance travelled and changes in

direction so as to prevent ourselves becoming disorientated. Such techniques may not be available to the novel user in a virtual world which they are unfamiliar, particularly if teleporting (e.g. instantaneously transferring between two different points in the world) is applied, as is the case in many gaming environments. Consequently, it is advantageous to provide an automated pathfinding system for the inexperienced user to help them navigate them through the virtual world, allowing them to overcome the problems of lack of familiarity and any frustrations with the traditional interface of self-navigation. Consequently, many systems have been developed which automatically calculate the best path through a virtual world, for example to find the shortest path to view a particular object or to pass through a room.

PRIOR ART

Article entitled "Smart Moves: Intelligent Pathfinding" by Bryan Stout, published in Game Developer, October 1996, and also available online at http://www.gamasutra.com/features/19970801/pathfinding.htm provides an broad introduction to various pathfinding techniques which can be used in virtual worlds. In particular, it discusses and compares breadth-first pathfinding (including Dijkstra's Algorithm), depth-first and best-first pathfinding techniques.

All pathfinding algorithms work by expanding possible paths, one node at a time until a path from the start node to the destination node is found. The order in which the nodes are attempted is critical to the speed and quality (e.g. length) of the final path that is found. In a breadth-first techniques, the algorithm searches firstly through all a nodes' immediate neighbours, and then moving onto the neighbours of those neigbouring nodes. In contrast, in depth-first techniques, the algorithm searches for a path recursively using a child node each time until reaching a predetermined depth, at which point it retraces it's steps and tries other child nodes. A best-first search chooses paths preferentially on the basis of an estimate of the shortest remaining distance to the goal.

The document discusses the various merits and disadvantages of search algorithms, including the preferred search algorithm known as A*, which provides an optimal method (both in terms of processing requirements and quality of path) for many types of pathfinding problems. The document proposes different techniques for overcoming some of the disadvantages in the A* algorithm. These include how to select an estimate of the remaining distance to the goal, and how the processing requirements can be reduced by splitting the virtual world into different regions (which can then be calculated separately) and using more efficient data structures.

The article entitled "Middleware Solutions for Artificial Intelligence in Computer Games" (Chapter 3 – AI Techniques) by Jan-Harald Fredriksen of the Norwegian University of Science

and Technology, located at http://www.idi.ntnu.no/grupper/su/fordypningsprosjekt-2003/fordypning2003-Jan-Harald-Fredriksen.pdf is also concerned with pathfinding techniques in virtual worlds, and the use of the A* algorithm. This document considers the use of cost constraints for different terrains to be taken into account when calculating a path through the virtual world, and the use of hierarchical pathfinding (finding a broad overall route before computing the detailed portions) and portals (which split regions of the virtual world into smaller parts) for improved computation performance.

The underlying structure of the virtual world on which the pathfinding in this document is based has two alternative versions. The first is referred to as the POV (points of visibility) approach, in which a plurality of nodes (called waypoints) are introduced into the virtual world, and which the path is found on the basis of line-of-sight links between the nodes. The second approach is the use of a navigation mesh in which a plurality of convex polygons cover the surface of the world, and in this case the pathfinding algorithm considers each polygon to be a node and calculates a path through the virtual world accordingly.

SUMARY OF THE INVENTION

According to the invention, there is provided ... [to be completed]

BRIEF DESCRIPTION OF THE DRAWINGS

For a better understanding of the present invention, specific embodiments will now be described, by way of example, with reference to the acompanying drawings, in which:
Figure 1 shows a schematic of a system according to an embodiment of the invention;
Figure 2 shows a representation of a virtual world environment;
Figures 3 to 6 show further representations of the virtual world environment of Figure 2; and
Figure 7 shows a flow chart of the method according to an embodiment of the invention, for processing points within a virtual world.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

Figure 1 illustrates a system 1 for creating, and allowing a user to navigate within, a virtual world. The system comprises a server computer 2, accessible to a content provider (a human user who will input the specification for the virtual world) via terminal 3, and accesible to a user (desiring to view and navigate through the virtual world) via another terminal 4. In this particular embodiment, access from both terminals to the server 2 is through a general network 5, such as for example the internet, although it is understood that the illustrated computer arrangement may be replaced by any suitable alternative which allows a content provider to design and

implement a virtual world, which a user can subsequently access and interact with.

A schematic of a simple virtual world environment is shown in Figure 2. This represents a room 20, including walls 21, doors 22, and various obstacles 23, 24 and 25 within the room. When specifying the design of this virtual world 20, the content provider inputs data in the form of co-ordinates representing the shape of the environment into the server 2 via terminal 3. The data might include co-ordinates representing a collisionable mesh defining the obstacles 23, 24, and 25 through which a navigating user is not allowed to pass.

Feature Nodes

The content provider is able to define features of interest within the room that a user may wish to view. For example, if the virtual world represents a tour through a historic building, it may be desirable to insert features such as statues, furniture, paintings and information boards. These may be represented by objects within the room, for example statue object 26, painting 27, and information boards (obstacle 25).

In order that these features may be visited by a user, the content provider defines feature nodes 28A, 28B and 28C (shown in Figure 3) associated with the features, which the user can view or interact with. A feature node is therefore essentially a point in the virtual world, linked either to an object or to the collisionable mesh, and in the embodiment has the following data set:

Node ID
Feature Group ID
Object ID of the linked mesh / object
XYZ of object pivot point data
Sphere of influence settings
Display settings

Node ID – an identifier for the node (e.g. text string "monet_painting") [?or is nodeID a unique numerical value?], which may be inherited from the object

Feature Group ID – an identifier for the type of feature associated with the node, for example all paintings may be in a first Feature Group identified as "paintings", whilst information boards might be a second Feature Group "information_points". The choice of groupings is highly dependent on the content provider and the type of features they wish to distinguish within the virtual world.

Object ID of linked mesh / object – this is the identifier for the object or collisionable mesh associated with the feature node, and may be the same as the node ID (e.g. "monet_painting")

XYZ of Object Pivot Point Data – these are the co-ordinates giving the location of the feature

node (and may be obtained by extraction from properties of the object / mesh with which the feature node is associated)

Sphere of influence settings – these define (for a 3D world) a three dimensional sphere (for example, illustrated as circle 31 in Figure 3) surrounding the feature node, and are used during the calculation of navigation paths through the virtual world (see later)

Display settings – these are the settings to be used for text / images assocated with the feature when constrained

Navigational Nodes

The content provider also defines initially a plurality of navigational nodes (29A, 29B, 29C, etc) within the virtual world, as shown in Figure 3. Essentially, these provide a simple matrix of navigational points which the automated path generation algorithm can use to plot a route through the environment. Associated with each navigational node is the following data set:

Node ID
Node Level ID (optional)
XYZ position relative to world
LoS data
LoS constraint settings (optional)
FoV constraint settings (optional)
Orientation constraint settings (optional)

Node ID – an identifier for the node is this unique for every node?

Node Level ID – this is an identifier which allows a level for the node to be defined, effectively grouping the nodes into different subsets, and is used during the path finding process to split the virtual world into different regions (which might be overlapping) for processing efficiency

XYZ position relative to the world – these are the x, y, z co-ordinates giving the location of the navigational node

LoS data – this data field will be empty initially, but is later used to contain a list of all nodes which the present node has Line of Sight (LoS) to, i.e. all the nodes that it is able to link to (see later for further details)

LoS constraint settings – the content producer may choose to manually define an angular LoS constraint (e.g. an angle range, in which the node is only allowed to search for nodes to link to if they fall within that viewing angle, see later for more details)

FoV constraint settings – the content producer may choose to define angular constraints for the Field of View (FoV) which will constrain the range of images allowed to be displayed to a user at that node (e.g. for node 29F, the FoV may be constrained in the angular direction towards statue object 28A to ensure that the user sees the object)

Orientation constraint settings – for some navigational nodes, the content producer may choose to constrain the allowed orientation of the user as they approach the node, by defining orientation constraint settings this forces the calculated path to approach the node from a particular direction.

Processing

The subsequent processing is explained with reference to the flow diagram in Figure 7. After node data has been input to the system (step 40), a node matrix is automatically created for the virtual world (step 41). This identifies for each navigational node all the other navigational nodes it is able to link to. Node linkages are calculated following basic roles of LoS (line of sight) for two nodes. This takes into account the collisionable mesh and any other objects deemed to crop the LoS of the nodes (i.e. no links are allowed through obstructions because a path between nodes cannot pass through these). Figure 4 illustrates all possible links for navigational nodes in the virtual world 20 (i.e. the lines linking nodes 29A, 29B, etc). For example, Node 29A has direct line of sight to both nodes 29B and 29F, but not to 29C because this would pass through the obstacle 23. A node matrix for virtual world 20 of Figure 4 is represented in tabular form below:

Node	Links to other nodes
29A	29B, 29F
29B	29A, 29C, 29D, 29E
29C	29B, 29D, 29F
29D	29B, 29C, 29E
29E	29B, 29D, 29F, 29G
29F	29A, 29C, 29E
29G	29E

At this stage each navigational node's data set is provided with LoS data on all its visible partners, including their XYZ position. Any navigational node which does not have line of sight to any other navigational nodes is deemed invalid and may be deleted from the system.

If the content provider had specified any LoS constraints for particular navigational nodes, then this would crop the angle it is allowed to search for node linkages. This is controlled using "LoS constraint settings" for the node, comprising degree values which slice the otherwise allowed 360 degree LoS. In this case, the node matrix above would differ as some of the links shown there would not be allowed. This functionality allows the content provider to create channels of nodes that ignore other nodes around them, and to produce breaks in the node matrix where nodes are not desired.

The next stage of processing involves the automatic creation of a more dense matrix of nodes (step 42), by subdividing the LoS linkages between the existing navigational nodes, to create new navigational nodes (also referred to as subdivided nodes) at each midpoint (step 43). Figure 5 shows these subdivided nodes (30A, 30B, 30C,... etc). After the new nodes have been added, the LoS data for each navigational node and each newly created subdivided node is updated to take account of all the multiple new possible LoS linkages (step 47). Figure 6 illustrates the newly created dense matrix with many more possible linkages between the nodes, where every node has a link to all the other nodes which it has LoS to. is this correct, or are the nodes only connected to the nearest ones? Even with only one cycle of subdivision, the number of nodes now existing has more than doubled, and the connectivity between the nodes has greatly increased. For example node 29D previously only had links to 29B, 29C and 29E, but is now additionally linked to 30B, 30C, 30D, 30E and 30F.

Further cycles of subdivision (step 42) may be repeated, until a predefined density of nodes is reached. This density may be defined in advance by the content provider (e.g. by specifying a minimum distance allowed between nodes), and on the creation of each new subdivided node (step 43), the system checks whether the new node will be too close to any other nodes (step 44). If so, the newly created node is deleted (step 45). Once new nodes are no longer being created (step 48), the subdivision process terminates (step 49). [Inventors to provide further details on the criteria used to determine whether the node density limit is reached / also maybe different node densities allowed for different areas?] The system has now created a suitably dense matrix of nodes which map out the non-collisionable environment in the virtual world. Advantageously, during the subsequent pathfinding calculation, there is no need to calculate collision data in real time because the matrix nodes are all constrained to navigable areas. In addition, with the higher density of nodes in the environment, the system can generate far more efficient paths than with the originally defined node matrix, and which enhance the user experience by giving a much smoother path through the virtual world.

Pathfinding

As mentioned earlier, the A* algorithm is an optimal pathfinding algorithm both in terms of the processing time required, and the quality of the path found. Start and destination nodes are defined in advance, and the algorithm explores multiple different paths to find the best route from the start to the destination. Consider the case of a user who wishes to tour through virtual world 20, entering at door 22, and who has indicated via a suitable interface that he desires to approach information board 25.

The start node is defined as node 29A, and the destination node as 29G. The A* algorithm

operates by maintaining two lists of nodes: a list of nodes that have already been explored (the Closed list) and a list of nodes linked to the ones that have been explored but have yet to be explored themselves (the Open list). The algorithm takes a node from the Open list, and if it is the destination node then the algorithm terminates. If not, all the nodes linked to the chosen node are added to the Open list. The A* algorithm is known as a heuristic method because it uses estimates to guide the search for nodes. In particular, in order to decide which node should be selected next from the Open list, all the nodes have an evaluation function f = cost + heuristic, and the node which is selected as the next current node from the Open list is the one with the lowest value for f.

The cost is a measure of the quality of the path so far from the start node to the current node (ie for a simple A* algorithm this could be the sum of the lengths of all the links in the path between the start node and the node). The heuristic function is an estimate of the cost to get from the current node to the destination node. The results of the A* algorithm are very much dependent on the choice of cost and heuristic functions, and it is the specific weighting and cost values used in the evaluation function for the embodiment of the invention that allow the optimum path to be generated.

The cost function of the embodiment takes into account various factors of the route, together with any user specific preferences. The cost function is a weighted sum of the following form:

$$cost = w1 * distance + w2 * interest + w3 * angle$$

where distance is length of the links so far, and w1 is the weighting ascribed to the current user indicating how important it is to them that the path is as short as possible. Interest gives a measure of how interesting the node (s) so far have been (but since we are looking to minimise the cost function, the lower the interest value the more interesting the node). Thus, feature nodes (typically of great interest to a user) may be given a value 0, whilst navigational nodes have interest value of 1 and therefore a higher cost. Weighting w2 gives a measure of how important it is to the user that they visit features of interest. Finally, angle gives a measure of how much the path so far wiggles, since this represents a sum (or average) of how much the angle between the entry and exit links for each node deviates from a straight line. Weighting w3 gives a measure of how important it is that for the user that the path wiggles as little as possible.

The heuristic is an estimate of the cost to get from the current node to the destination node, and a possible choice for this may be:

heuristic = w1 * euclidian_distance_to_destination + w2 * estimated_nodes_to_destination

inventors to confirm if this is correct / acceptable example..?

The first term includes euclidian_distance_to_destination which is a straight line from the current node to the destination node, weighted by w1. The second term assumes that all the nodes will be boring navigational nodes (and therefore each have a cost associated with them), and provides some estimate of how many nodes are required before the destination is reached, for example estimated_nodes_to_destination could be calculated by dividing the euclidian distance by the average length of a link between nodes.

The system therefore runs through the A* algorithm for the nodes using the evaluation function, until the final destination node is reached. Importantly, the weightings w1, w2, and w3 can be varied depending on the user profile.

Defining a user profile

A suitable interface (not shown) allows the user to define their profile (e.g. what they are particularly interested in, and how they like to navigate through the virtual world). The user profile settings are then incorporated into the weightings above used for the A* algorithm, which consequently will result in different paths being generated for different users. The translation of the user interests into weightings allow different routes to be compared against each other such that the optimum path for that user can be generated. It can be seen that for a user who is particularly interested in visiting features of interest (paintings, landmarks, etc) it is likely to be the case that the system will not calculate the quickest or shortest route between two points. Rather, the lowest cost path (i.e the best route for that user) will be one which visits as many features of interest as possible.

The user may input their profile upon first entering the virtual world. Some options include:

Always select shortest path	on / off
Always select quickest path	on / off
Ignore all features	on / off
Gaze orientation	on / off
Velocity control	on / off
Importance of path being short	weighting
Importance of path not being wiggledy	weighting
Importance of visiting Feature Group 1 [e.g. paintings]	weighting
Importance of visiting Feature Group 2 [e.g. information points]	weighting
Importance of visiting Feature Group 3 [e.g. landmarks]	weighting

The first five options are overrides, and are thus non-weighted. For example, if the user selected "Always select shortest path" as ON, then the weighting w2 for the features of interest become irrelevant and will be set to zero.

The final five options are all weighted, and however the weightings are distributed they should all be normalised (e.g. add up to 100% in total). Also, two different Feature Groups should not be allowed to have the same weighting why is this not allowed?. Any option which is not important to the user and not given a weighting is deemed irrelevant and not taken into account in the pathfinding process.

Dynamic Navigational Nodes

In addition to the fixed navigational nodes (29A, 29B, 29C, etc), defined initially by the content provider, the system may also include dynamic navigational nodes (not shown). These nodes operate like navigational nodes in the sense that they can form part of the path generated for the user, but provide enhanced functionality because the user is able to move them. A dynamic navigational node is associated with an interactive mesh object in the virtual world environment that can be moved around by the user. The user is able to specify that they want the dynamic navigational node to be a destination point, and the path will be calculated to it's location.

If a dynamic navigational node is placed a large distance from other navigational nodes, then the subdivision process described earlier may be used to fill in new nodes until a suitable node density is achieved in that area. This then allows a smooth path to be calculated to the node.

If a dynamic navigational node is placed without a line of sight to any other navigational nodes (i.e. in a blind spot) then the node is defined as invalid – lost from the Node Matrix (although still present in the environment) and the user is prevented from selecting it as a destination. Alternatively, the system may prevent the user from placing the node in a blind spot, and only allows it to be moved into valid positions.

Switch Navigational Nodes

In addition to the fixed navigational nodes and the dynamic navigational nodes, the system may also include switch navigational nodes (not shown), defined by the content producer. As mentioned earlier, navigational nodes have a Level ID, specifying which level the node belongs to, and effectively grouping the nodes into different subsets for more efficient path calculation. For example, if the virtual world consists of two different rooms, then the navigational nodes in one room may be defined as belonging to one level, and the navigational nodes in the second

room to a different level. During the pathfinding process in one room, only the nodes of that level are evaluated, saving takes into account nodes on a different level. The switch nodes then provide the functionality to swap from one level to the second level (e.g. the switch navigational node might be positioned in a doorway between the two rooms).

Associated with each switch navigational node is the following data set:

Node ID
Node Level ID #1
Node Level ID #2
XYZ position relative to world
LoS data
Switching data

Node ID – an identifier for the node

Node Level ID #1 – the identifier for the first level

Node Level ID #2 – the identifier for the second level

XYZ position relative to the world – these are the x, y, z co-ordinates giving the location of the switch navigational node

LoS data – this gives the data for all nodes (for both levels) which the present node has Line of Sight (LoS) to

Switching data – gives the criteria of when the generated path can / cannot switch from navigational nodes on one level to nodes on the other level

Event Feature Nodes

As described earlier, the content provider is able to define feature nodes (28A, 28B, 28C, etc) associated with the features of interest in a virtual world. The content producer is also able to define event feature nodes (not shown) which have the extra functionality that they react to users within the virtual world, or are only active at certain predefined times.

These event feature nodes expand the functionality of the system into the gaming and interactive online environment platforms, where multiple users can interact with the environment and with each other (e.g. the Event Feature may simulate an explosion, or the opening and closing of an interactive chat-room / race event. The content provider must specify when the event feature will be active / inactive, or what triggers it will respond to. An example data set is as follows:

Node ID
Feature Group ID
Trigger data
XYZ of pivot point data
Sphere of influence settings
Time settings
Display settings

Node ID – an identifier for the node (e.g. text string "explosion")

Feature Group ID – an identifier for the type of feature associated with the node

Trigger data – the conditions which will cause the event feature node to become active

XYZ of Pivot Point Data – these are the co-ordinates giving the location of the event feature node

Sphere of influence settings – these define (for a 3D world) a three dimensional sphere surrounding the feature node. However, in addition to the settings used for the standard feature nodes, these additionally include a time dependant feature which may allow the sphere of influence to degrade over time.

Time settings – these define when the event feature node will be active / inactive

Display settings – these are the settings to be used for text / images assocated with the feature when constrained

Appendix II

PATHFINDING SYSTEM

TECHNICAL FIELD

The invention relates to a method and system for determining a path along a plurality of points in a representation of a real or virtual world. In particular, but not exclusively, the invention relates to a method and system for determining a path in which for each node of the path a constraint is imposed on the entry and/or exit angle made by each LoS connection to adjacent nodes along the path.

INTRODUCTION

As more and more systems are becoming available for determining an optimal path between a set of points it is becoming more and more important for the paths which are generated to be user friendly as well as optimal in terms of efficiency. System may provide representations of both virtual environments (for example, where such systems can be implemented as games or other virtual world environments) or representations of real environments (for example, where such systems can comprise navigational aids for both sea and land environments),

Systems in which 3-dimensional virtual reality environments (virtual worlds) are defined and created are becoming increasingly prevalent in the current technological age. Such computer systems are applicable in numerous commercial applications, such as for online tourist sites, fantasy worlds, gaming, architectural walk-throughs, estate agency (virtual tours of houses for sale), and many others. Typically, a virtual world might be defined by multiple x,y,z, co-ordinate sets which together map out an environment in three dimensions (x, y, and z axes). Objects and walls (or other items which exist within the virtual world), which a user navigating through the virtual world is not allowed to walk through, can be implemented by what is termed a collisionable mesh. The collisionable mesh may therefore comprise a further set of three dimensional co-ordinates defining shapes within the virtual world which the user must navigate around.

Navigation is required within virtual worlds for any entity passing through that world, such as a computer generated person, vehicle or gaming character. This is typically under the control of an external human user. Traditionally, many commercial implementations of virtual environments have been built on the basis of the self-navigation techniques inherited from games engines, where exploration of all areas of an unknown virtual world is expected, and also a high degree of user competency is assumed. However, as the application of virtual worlds

spreads into other commercial areas, it is important that the functionality of navigation evolves to suit the competency and requirements of a whole new wider group of users. Such users may not be as proficient in the navigation around virtual worlds, and furthermore may be disinclined to spend large amounts of time familiarising themselves with the environment. They are likely to become disorientated or bored much more quickly, or even miss important content entirely, unless the user experience is significantly improved.

In our day-to-day lives, humans typically employ various techniques to help us with our navigation when moving from place to place. We retain an idea of our location and orientation using fixed points of reference (landmarks), measures of distance travelled and changes in direction so as to prevent ourselves becoming disorientated. Such techniques may not be available to the novel user in a virtual world which they are unfamiliar, particularly if teleporting (e.g. instantaneously transferring between two different points in the world) is applied, as is the case in many gaming environments. Consequently, it is advantageous to provide an automated pathfinding system for the inexperienced user to help them navigate them through the virtual world, allowing them to overcome the problems of lack of familiarity and any frustrations with the traditional interface of self-navigation. Consequently, many systems have been developed which automatically calculate the best path through a virtual world, for example to find the shortest path to view a particular object or to pass through a room.

However, even when pathfinding is automated and a guided path is provided for a user to automatically navigate along, a user may become confused when many changes of direction occur. These may be actual changes in direction of the guided path, or changes in direction of the field of view of the user as they are navigating along the guided path.

The result in both cases is for the user to become more disorientated, and to lose their sense of direction. Accordingly, it is useful if the guided path along which a user is automatically navigated can be constrained so that the if, when navigating along a path automatically generated from A to B, the path changes direction several times. As the user becomes more familiar, they may wish to follow a more winding path, as they will still maintain their sense of orientation. Accordingly, it is useful if the angle differential between the mesh points used to generate the path can be constrained according to a user's personal preference.

Nodal path generation is well known, for example, "Smart Moves: Intelligent Pathfinding" by Bryan Stout, published in Game Developer, October 1996, and also available online at http://www.gamasutra.com/features/19970801/pathfinding.htm provides an broad introduction to various pathfinding techniques which can be used in virtual worlds. In particular, it discusses and compares breadth-first pathfinding (including Dijkstra's Algorithm), depth-first and best-first pathfinding techniques.

All pathfinding algorithms work by expanding possible paths, one node at a time until a path from the start node to the destination node is found. The order in which the nodes are attempted is critical to the speed and quality (e.g. length) of the final path that is found. In a breadth-first techniques, the algorithm searches firstly through all a nodes' immediate neighbours, and then moving onto the neighbours of those neighbouring nodes. In contrast, in depth-first techniques, the algorithm searches for a path recursively using a child node each time until reaching a predetermined depth, at which point it retraces it's steps and tries other child nodes. A best-first search chooses paths preferentially on the basis of an estimate of the shortest remaining distance to the goal. However, Stout does not discuss the problems associated with constraining the angle differential between nodes.

The article entitled "Middleware Solutions for Artificial Intelligence in Computer Games" (Chapter 3 – AI Techniques) by Jan-Harald Fredriksen of the Norwegian University of Science and Technology, located at http://www.idi.ntnu.no/grupper/su/fordypningsprosjekt-2003/fordypning2003-Jan-Harald-Fredriksen.pdf is also concerned with pathfinding techniques in virtual worlds, and the use of the A* algorithm. This document considers the use of cost constraints for different terrains to be taken into account when calculating a path through the virtual world, and the use of hierarchical pathfinding (finding a broad overall route before computing the detailed portions) and portals (which split regions of the virtual world into smaller parts) for improved computation performance. . However, Fredriksen does not either consider applying constraints to the path generation such as angle-differential. Fredriksen describes two alternative versions for the underlying structure of the virtual world on which the pathfinding in this document is based. The first is referred to as the POV (points of visibility) approach, in which a plurality of nodes (called waypoints) are introduced into the virtual world, and which the path is found on the basis of line-of-sight links between the nodes. The second approach is the use of a navigation mesh in which a plurality of convex polygons cover the surface of the world, and in this case the pathfinding algorithm considers each polygon to be a node and calculates a path through the virtual world accordingly.

SUMMARY OF THE INVENTION

According to a first aspect of the present invention, there is provided a method of determining a path along some of a plurality of points in a representation of an environment, the representation comprising a plurality of initially defined points, and obstructions through which the path cannot pass, the method comprising: defining a plurality of additional points by repeating the following steps: defining a new point located between a first existing point and a second existing point, wherein a first line between the first point and the new point and a second line between the new point and the second point do not pass through any of the obstructions; and calculating the path based on any combination of new and/or initially defined points,

wherein the angle of deviation of the first line to a line between the first and second points is constrained.

The constraint may be imposed by assigning a varying angle of deviation cost value in dependence on the angle to which the first line deviates from the second line.

The angle of deviation cost value may be zero when said angle is 180 degrees, and said angle of deviation cost value is a maximum when said angle is 360 degrees.

Alternatively, an alternative system of reference may be used to define the angle of deviation, in which case the cost value may be higher when the angle is 180 degrees, and lower when it is zero degrees.
The environment may comprise a virtual world.

The representation may model a real world navigable environment. The navigable environment may comprise a sea-based environment and/or a land-based environment.

The angle of deviation may be constrained to a predetermined range. The range of the angle of deviation may be determined as a function of a range of angles within which the first line approaches the new point from the first point, and a predetermine angle of deviation.

The method may be performed to create the content of the virtual environment and/or the method may be performed after the content creation of the virtual environment to dynamically determine the path.

Advantageously, if the nodal subdivision process of the invention is performed at least partially prior to generating a spline path from a start node to a destination node, the processing required to generate the spline path is reduced. For example, if a node is static within the virtual environment, sub-divisions of the nodal space around that node may be advantageously performed as a preprocessing stage prior to generating a path. However, if the virtual environment is populated with may nodes which are dynamically changing such a preprocessing stage may provide a smaller performance gain.

The path determined may have a dynamically changing destination node. The path determined may define a field of view of said virtual environment.

A second aspect of the invention seeks to provide a system for determining a path along some of a plurality of points in a representation of an environment, the representation comprising a plurality of initially defined points, and obstructions through which the path cannot pass, the

system comprising: processing means to define a plurality of additional points by repeatedly defining a new point located between a first existing point and a second existing point, wherein a first line between the first point and the new point and a second line between the new point and the second point do not pass through any of the obstructions; and processing means to calculate the path based on any combination of new and/or initially defined points, wherein the angle of deviation of the first line to a line between the first and second points is constrained.

The system may further comprise means to enable a user to be automatically navigated along said path.

The system may comprise means to enable a user to select the degree to which said path is wiggly.

A third aspect of the invention provides a user interface for a system according to the second aspect, wherein the user interface is arranged to provide said means to enable the user to select said degree to which said path is wiggly.

The user interface may be provided to implement content creation of the virtual world and/or implement a user profile for said virtual environment.

A fourth aspect of the invention provides a storage medium carrying computer readable code representing instructions for causing one or more processors to perform the method of the first aspect when the instructions are executed by the processor or processors.

A fifth aspect of the invention provides a computer program comprising instructions for causing one or more processors to perform the method according to the method of the first aspect when the instructions are executed by the processor or processors.

A sixth aspect of the invention provides a computer data signal embodied in a carrier wave and representing instructions for causing one or more processors to perform the method according to any of the first aspects when the instructions are executed by the processor or processors.

A seventh aspect of the invention provides a storage medium carrying computer readable code representing instructions for causing one or more processors to operate as the system according to the second aspect when the instructions are executed by the processor or processors.

An eighth aspect of the invention provides a computer program comprising instructions for causing one or more processors to operate as the system according to the second aspect of the

invention when the instructions are executed by the processor or processors.

A ninth computer data signal embodied in a carrier wave and representing instructions for causing one or more processors to operate as the system according to the second aspect of the invention when the instructions are executed by the processor or processors.

This invention is applicable to any type of virtual world, which might include for example fantasy worlds or worlds based upon a real environment such as a historic building or design for a future building, or a house that is for sale, and in which the obstructions might comprise walls or other obstacles, for example. The representation in the sense used above is likely to refer to stored digital data that represents such a world. Alternatively, the virtual environment may provide a representation of the real world.

The new point may be located on the mid point of the link. Step (a) may further comprise deleting the new point if it is less than a predefined distance from another of the points, and the predefined distance may vary in different regions of the virtual world. The method may further comprise identifying a link as not being suitable for providing the location for a new point if said link intersects another one of said links which is shorter. A new point may be deleted if it does not have line of sight to each of a pair of points which do not have line of sight to each other and/or may be deleted if it does not form part of a path between two other nodes that is shorter than the shortest path which would exist between said two points without said new point.

Advantageously, the arrangements of the invention employ a technique which allows a content provider to define some initial points for navigation in a virtual world, allowing the content provider to influence the routes taken and the items of interest that can be visited by a user. The system then automatically multiplies the number of points in a manner which will improve the user's experience along the subsequently generated path through the environment compared with only a limited set of originally defined navigation points.

In comparison with previously known systems in which a path through a virtual world is not constrained to specific navigation points, the embodiments of the invention provide improved processing during the actual pathfinding procedure. This is because no collision detection (ensuring the path avoids obstacles, etc) is required during pathfinding in the embodiments since all the new navigable nodes and links have been added on line of sight principles, and therefore by definition already avoid all the potential obstacles.

Additionally, these embodiments provide advantages over previous known systems such as the AA Routefinder which are necessarily limited to finding routes along specified predefined roads, etc. In contrast, in the embodiments, the user is able to navigate within all areas of the virtual

world.

The aspects of the invention are as set out in the accompanying claims, together with dependent claims representing preferred features of the invention. The preferred features of the invention may be suitably adapted to be combined with each other and/or with any of the aspects of the invention as is apparent to those skilled in the art.

BRIEF DESCRIPTION OF THE DRAWINGS

For a better understanding of the present invention, specific embodiments will now be described, by way of example, with reference to the accompanying drawings, in which:

Figure 1 shows a schematic of a navigation system according to an embodiment of the invention;

Figure 2 shows a diagrammatic representation of a virtual world environment;

Figures 3 to 12 show further diagrammatic representations of the virtual world environment of Figure 2; and

Figure 13 shows a flow chart of the method for processing points within a virtual world according to one embodiment of the invention;

Figures 23A to 14C show how an angle differential weighting influences the connection angle between two nodes in a virtual world according to one embodiment of the invention;

Figures 24A and 24B show two wiggly paths with differing degrees of waviness.

Figure 25A shows a navigational node with a valid LoS Angle range constraint;

Figure 25B shows how range of LoS angles may implement a constraint on the connection angel between two nodes according to another embodiment of the invention; and

Figure 26 shows how a range of LoS angles may implement a constraint on the angle of entry and angle of exit of an automatically generated path passing through a node according to another embodiment of the invention.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

The best mode of the invention currently contemplated by the inventors will now be described with reference to the accompanying drawings.

Figure 1 illustrates a system 1 for creating, and allowing a user to navigate within, a virtual world. The system comprises a server computer 2, accessible to a content provider (a human user who will input the specification for the virtual world) via terminal 3, and accessible to a user (desiring to view and navigate through the virtual world) via another terminal 4. In this particular embodiment, access from both terminals to the server 2 is through a general network 5, such as for example the internet, although it is understood that the illustrated computer arrangement may be replaced by any suitable alternative which allows a content provider to design and implement a virtual world, which a user can subsequently access and interact with.

A schematic of a simple virtual world environment is shown in Figure 2. This represents a room 20, including walls 21, doors 22, and various obstacles 23, 24 and 25 within the room. When specifying the design of this virtual world 20, the content provider inputs data in the form of co-ordinates representing the shape of the environment into the server 2 via terminal 3. The data might include co-ordinates representing a collisionable mesh defining the obstacles 23, 24, and 25 through which a navigating user is not allowed to pass.

Feature Nodes

The content provider is able to define features of interest within the room that a user may wish to view. For example, if the virtual world represents a tour through a historic building, it may be desirable to insert features such as statues, furniture, paintings and information boards. These may be represented by objects within the room, for example statue object 26, painting 27, and information boards (obstacle 25).

In order that these features may be visited by a user, the content provider defines feature nodes 28A, 28B and 28C (shown in Figure 3) associated with the features, which the user can view or interact with. A feature node is therefore essentially a point in the virtual world, linked either to an object or to the collisionable mesh, and in the embodiment has the following data set:

Node ID
Feature Group ID
Object ID of the linked mesh / object
XYZ of object pivot point data
Sphere of influence settings
Display settings

Node ID – an unique identifier for the node (e.g. text string "monet_painting_01" or numerical identifier "00563")

Feature Group ID – an identifier for the type of feature associated with the node, for example all paintings may be in a first Feature Group identified as "paintings", whilst information boards might be a second Feature Group "information_points". The choice of groupings is highly dependent on the content provider and the type of features they wish to distinguish within the virtual world.

Object ID of linked mesh / object – this is the identifier for the object or collisionable mesh associated with the feature node, and may be the same as the node ID (e.g. "monet_painting_01")

XYZ of Object Pivot Point Data – these are the co-ordinates giving the location of the feature node (and may be obtained by extraction from properties of the object / mesh with which the feature node is associated)

Sphere of influence settings – these define (for a 3D world) a three dimensional sphere (for example, illustrated as circle 31 in Figure 3) surrounding the feature node, and are used during the calculation of navigation paths through the virtual world

Display settings – these are some of the settings to be used for text / images associated with the feature

Navigational Nodes

The content provider also initially defines a plurality of navigational nodes (29A, 29B, 29C, etc) within the virtual world, as shown in Figure 3. Essentially, these provide a simple matrix of navigational points which the automated path generation algorithm can use to plot a route through the environment. Associated with each navigational node is the following data set:

Node ID
Node Level ID (optional)
XYZ position
LoS data
LoS constraint settings (optional)
FoV constraint settings (optional)
Orientation constraint settings (optional)

Node ID – a unique identifier for the node

Node Level ID – this is an identifier which allows a level for the node to be defined, effectively grouping the nodes into different subsets, and used during the path finding process to split the virtual world into different regions for processing efficiency. For example, two different rooms in the virtual world could be represented by different regions of nodes, with the doorway between them comprising a switch node. For processing efficiency, a path can initially be calculated based on only the nodes of one of the rooms (thus saving erroneously calculating at this stage paths which enter the second room, and only later allowing the pathfinding calculation to extend to the second room via the switch node). Additionally it is possible that the different sub-sets could cover overlapping regions (for example a first level of nodes might provide coarse coverage of a large area while different levels might provide finer coverage of the same area or subsets of that area).

XYZ position– these are the x, y, z co-ordinates giving the location of the navigational node

LoS data – this data field will be empty initially, but is later used to contain a list of all nodes which the present node has Line of Sight (LoS) to, i.e. all the nodes that it is able to link to (see later for further details)

LoS constraint settings – the content producer may choose to manually define an angular LoS constraint (e.g. an angle range, in which the node is only allowed to search for nodes to link to if they fall within that viewing angle, see later for more details)

FoV constraint settings – the content producer may choose to define angular constraints for the Field of View (FoV) which will constrain the range of images allowed to be displayed to a user at that node (e.g. for node 29F, the FoV may be constrained in the angular direction towards statue object 28A to ensure that the user sees the object)

Orientation constraint settings – for some navigational nodes, the content producer may choose to constrain the allowed orientation of the user as they approach the node, by defining orientation constraint settings this forces the calculated path to approach the node from a particular direction.

The content provider is thus able, through the use of navigational nodes, to define key positions within the virtual world from which the potential user is able to best experience the features of interest within the environment. The content provider can define the exact position, direction of approach (orientation settings), and viewing angle (FoV constraint settings) so as to increase the quality of the user's experience of the features of interest, compared with other systems which are completely automated.

Processing

The subsequent processing is explained with reference to the flow diagram in Figure 13. After node data has been input to the system (step 40), a node matrix is automatically created for the

virtual world (step 41). This identifies for each navigational node all the other navigational nodes it is able to link to. Node links are calculated following basic rules of LoS (line of sight) for two nodes. This takes into account the collisionable mesh and any other objects deemed to crop the LoS of the nodes (i.e. no links are allowed to pass through obstructions because a path between nodes cannot pass through these). Figure 4 illustrates all possible LoS links for navigational nodes in the virtual world 20 (i.e. the lines linking nodes 29A, 29B, etc). For example, Node 29A has direct line of sight to both nodes 29B and 29F, but not to 29C because this would pass through the obstacle 23. A node matrix for virtual world 20 of Figure 4 is represented in tabular form below:

Node	Links to other nodes
29A	29B, 29F
29B	29A, 29C, 29D, 29E
29C	29B, 29D, 29F
29D	29B, 29C, 29E
29E	29B, 29D, 29F, 29G
29F	29A, 29C, 29E
29G	29E

At this stage each navigational node's data set is provided with LoS data on all its visible partners, including their XYZ position. Any navigational node which does not have line of sight to any other navigational nodes is deemed invalid and may be deleted from the system.

If the content provider had specified any LoS constraints for particular navigational nodes, then this would crop the angle it is allowed to search for node links. This is controlled using "LoS constraint settings" for the node, comprising degree values which slice the otherwise allowed 360 degree LoS. In this case, the node matrix above would differ as some of the links shown there would not be allowed. This functionality allows the content provider to create channels of nodes that ignore other nodes around them, and to produce breaks in the node matrix where nodes are not desired. This can improve the user experience because in this manner the content provider can influence the routes taken by a user within the virtual world.

The next stages of processing will add in additional nodes on the basis of the LoS links already defined. However, before that occurs, the system checks all the links (step 42) to decide which links will / will not be used as a base for the new nodes. In particular, the system looks for any crossed links (such as the 29B-29E link which crosses the 29C-29D link), and marks the longest of the crossed links as not being available for use as a base for the new nodes. In this embodiment a data flag is associated with the link (in either the node matrix or the individual nodes' LoS link data) to indicate that this link is "not to be subdivided". The affected link is

shown as a dotted line in Figure 5.

The next stage of processing involves the automatic creation of a more dense matrix of nodes (step 43). This is achieved by subdividing the LoS links between the existing navigational nodes so as to add new navigational nodes (also referred to as subdivided nodes) at each midpoint (step 44). Figure 6 shows these subdivided nodes (30A, 30B, 30C,... etc). As more and more new nodes are added during each cycle through step 43, the density of nodes increases, up to a maximum allowed node density. This density may be predefined in advance by the content provider, for example by specifying a minimum allowed distance from a node to it's nearest neighbour. After each new node is added, the system checks whether the new node meets the density criteria (step 45). If it falls outside the allowed density criteria, the new node is deleted (step 46). Otherwise, the node is kept. Once every LoS link has been examined for subdivision (step 47), the system checks whether any new nodes were added during the last pass (step 48). If no new nodes were created during a pass, the system terminates (step 49).

For each pass, after the new nodes have been added, the LoS data for every node (both the navigational nodes and each newly created subdivided node) is updated to take account of all the multiple new possible LoS links (step 50). Figure 7 illustrates the newly created dense matrix with many more possible LoS links between the nodes.

The system now checks which of the newly added nodes are valid (step 51). A node is defined as valid if it links any two other nodes which do not otherwise have line of sight to each other. This is illustrated for node 30A in Figure 8. In this drawing, for clarity, the only nodes and links illustrated are node 30A and the seven other nodes it links to. It can clearly be seen that node 30A acts as a bridge for many different pairs of nodes (i.e. it forms a connection between two nodes which do not have LoS to each other), such as between nodes 29B and 30F. Therefore node 30A is valid, and in fact only requires that it is connected to one pair of nodes which do not have LoS, to be defined as a valid node. Any invalid nodes are deleted.

Figure 9 shows the same procedure for node 30B. Node 30B is defined as valid. However at this stage, any of 30Bs links which do not assist in bridging between two nodes are deleted. This is the case for the link between node 30B and 29C. Since node 29C already has line of sight links to all the nodes shown on Figure 9 (and reciprocally node 30B also has line of sight to all the nodes linked to 29C), therefore link 30B-29C cannot help to bridge between any pair of nodes that do not have line of sight to each other, and the link 30B-29C is deleted.

Link 30B-30C would also appear from Figure 9 to be redundant, because node 30C already has line of sight to all the nodes in Figure 9. However, link 30B-30C is actually valid because it bridges, for example, nodes 30B and 30A (which do not have line of sight to each other) which

can be seen from Figure 7, and therefore link 30B-30C would not be deleted.

Validity checks are carried out in this manner for all new nodes, and two others (30E and 30H) are shown, in Figures 10 and 11 respectively.

After the validity checks have been performed on the new nodes, the system again checks for any crossed links (step 52). These are the links which are not to be used in any subsequent cycles of subdivision. The links are identified as before, by looking for links which cross each other and flagging the longest ones as "not to be subdivided". This has been performed for all the links, and the resulting link matrix is illustrated in Figure 12 (where dotted lines illustrate those links which are not to be used in the future for subdivisions to add more nodes).

Thus one cycle of subdivision has been completed, and even after this, the number of nodes has more than doubled, and the connectivity between the nodes has greatly increased. For example node 29D previously only had links to 29B, 29C and 29E, but is now additionally linked to 30B, 30C, 30E and 30F.

Further cycles of subdivision (step 43) may be repeated, until the system has created a suitably dense matrix of nodes which map out the non-collisionable environment in the virtual world. Advantageously, during the subsequent pathfinding calculation, there is no need to calculate collision data in real time because the matrix nodes are all constrained to navigable areas. In addition, with the higher density of nodes in the environment, the system can generate far more efficient paths than with the originally defined node matrix, and which enhance the user experience by giving a much smoother path through the virtual world which typically follows the type of behaviour a user is used to when navigating a real world environment. Once the path has been calculated a spline is generated to further interpolate and hence to smooth the journey still further. In the present embodiment, once the spline has been generated along a particular path, a default speed along the path is set. In the present embodiment this is set to vary between a fairly fast speed in areas of little interest and to a relatively lower speed around features of interest. Ideally this is done by smoothly varying the speed to have minimum values at the feature nodes of most interest and to have maximum speeds in between these points.

Although in the embodiment above, the node density criteria is specified in terms of the minimum allowed distance between a new node and an already existing node, this could be replaced by any other suitable mechanism for specifying and testing the node density. Another possible method is to specify in advance a maximum allowed number of cycles of subdivision before the algorithm must terminate.

Although In the embodiment above, the check for whether a new node is valid involves checking whether the node acts as a bridge between two other nodes, an alternative/additional criteria which may be used to define whether each newly added node is valid is to check whether the new node assists in providing a shorter path between any pair of nodes.

Pathfinding

As mentioned earlier, the A* algorithm is an optimal pathfinding algorithm both in terms of the processing time required, and the quality of the path found. Start and destination nodes are defined in advance, and the algorithm explores multiple different paths to find the best route from the start to the destination. Consider the case of a user who wishes to tour through virtual world 20, entering at door 22, and who has indicated via a suitable interface that he desires to approach information board 25.

The start node is defined as node 29A, and the destination node as 29G. The A* algorithm operates by maintaining two lists of nodes: a list of nodes that have already been explored (the Closed list) and a list of nodes linked to the ones that have been explored but have yet to be explored themselves (the Open list). The algorithm takes a node from the Open list, and if it is the destination node then the algorithm terminates. If not, all the nodes linked to the chosen node are added to the Open list. The A* algorithm is known as a heuristic method because it uses estimates to guide the search for nodes. In particular, in order to decide which node should be selected next from the Open list, all the nodes have an evaluation function f = cost + heuristic, and the node which is selected as the next current node from the Open list is the one with the lowest value for f.

The cost is a measure of the quality of the path so far from the start node to the current node (i.e. for a simple A* algorithm this could be the sum of the lengths of all the links in the path between the start node and the node). The heuristic function is an estimate of the cost to get from the current node to the destination node. The results of the A* algorithm are very much dependent on the choice of cost and heuristic functions, and it is the specific weighting and cost values used in the evaluation function for the embodiment of the invention that allow the optimum path to be generated.

The cost function of the embodiment takes into account various factors of the route, together with any user specific preferences. The cost function is a weighted sum of the following form:

$$cost = w1 * distance + interest\_component + w3 * angle$$

where distance is length of the links so far, and w1 is the weighting ascribed to the current user indicating how important it is to them that the path is as short as possible. Interest_component

gives a measure of how interesting the node (s) so far have been (but since we are looking to minimise the cost function, the lower the interest_component value the more interesting the nodes so far have been). Thus, in the present embodiment a summation over the node or nodes included in the partial path built thus far of the form:

$$\sum_{PARTIAL\_PATH}(1 - w2 * Interest)$$

is used, in which w2 is the percentage weighting interest given to the feature group associated with the node in question, and Interest is a measure of how strongly a node is associated with the corresponding feature associated with the node. Thus, in the present embodiment, feature nodes (typically of great interest to a user) may be given an Interest value 1, whilst navigational nodes have interest value of 0 and therefore when summed as above give a higher cost. As mentioned, weighting w2 gives a measure of how important it is to the user that they visit features of interest of each respective feature group. Finally, angle gives a measure of how much the path so far wiggles, since this represents a sum (or average) of how much the angle between the entry and exit links for each node deviates from a straight line. Weighting w3 gives a measure of how important it is for the user that the path wiggles as little as possible (i.e. a zero value of w3 would suggest that the user did not mind how wiggledy the path was whilst a high value (e.g. 100%) would indicate that the user would prefer to avoid wiggledy paths). This is shown in more detail in Figures 23A to 23C of the accompanying drawings.

In Figures 23A, 23B, 23C a connection between nodes A and C is established via node B with varying degrees of efficiency. The LoS l_1 between nodes A and B defines an initial reference angle $\phi_1 = 0°$. The LoS l_2 between nodes B and C subtends angle ϕ_2 from reference angle ϕ_1.

Thus in Figure 23A, the LoS route between nodes A and B has $\Delta\phi = \phi_2 - \phi_1 = 0°$, which provides the maximum efficiency. In contrast, the connections shown in Figures 23B and 23C are less efficient. In Figure 23B, $\Delta\phi = \phi_2 - \phi_1 < 90°$, whereas in Figure 23C, $\Delta\phi = \phi_2 - \phi_1 = 0°$, $= \phi_2 - \phi_1 > 90°$. If the angle of differential is constrained to $\Delta\phi < 90°$, then the node connection shown in Figure 23B would be available, whereas that shown in Figure 23C would be inefficient.

According to one embodiment of the invention, the LoS angle differential $\Delta\phi$ between the entry LoS l_1 and the exit LoS l_2, provides an additional weighting for the pathfinding system described herein. The LoS angle differential can also be used to constrain pathfinding in other systems in which it is advantageous if connections between nodes are constrained according to whether a path is to be more or less "wiggly".

In the embodiments shown in Figures 23A to 23C, if the LoS angle differential value $\Delta\phi$ is

assigned a highest for $\Delta\phi = 0°$ and a $\Delta\phi = 180°$ is assigned a low rating, the path finding system will optimise straight connections between nodes A and C. If however, $\Delta\phi = 0°$ had the lowest rating and $\Delta\phi = 90°$, for example, the path finding system would find more bendy paths acceptable. This is shown schematically in Figures 24A and 24B.

In Figure 24A a relatively straight path is shown between navigational nodes A and C, whereas in Figure 24B, a more wiggly path between navigational nodes A and C is shown. The path shown in Figure 24A is representative of a weight being provided by the LoS angle differential which constrains the path generation to optimise LoS connections between the nodes along the path. In contrast, the path shown in Figure 24B is representative of a weight being provided by the LoS angle differential which is representative of a weight being provided by the LoS angle differential which constrains the path generation to generate a wiggly path between nodes A and C. Thus, in the examples shown in Figures 23A to 23C, as the angle between the first and second line increases, the efficiency of the nodal connection decreases. The rating for each node is processed for the entire path, for example, to determine an average rating for the path which will be dependent on the number of nodes along the path, to determine a line of sight (LoS) angle differential ranking value for each different nodal path between two points.

Whilst the angle differential LoS weighting may be used by a content designer, it is also useful to provide a means for a user of the navigation system to apply a constraint on the angle differential LoS, as this can facilitate the user's orientation as they navigate the virtual environment.

Accordingly, another embodiment of the invention provides a user interface arranged to implement an angle differential navigation constraint. The user interface enables a user to constrain the LoS angle differential ranking value of the path in dependence on one or more preferences determined by the user. This enables a straighter guided path to be provided along which the user is automatically navigated according to the pathfinding system employed. This can enhance the user's experience as the user is able to retain more of a sense of direction. Moreover, by constraining the angle differential of the LoS of the guided path along which the user is automatically navigated by the system, the FoV that the user can also be constrained to enable the user to be less disorientated by the number and approach taken towards certain feature nodes. If more bendy automatic path navigation is selected by the user, then the more bendy path would enable the user to see more features from different directions possible, and even loop around certain features. This may be advantageous if a user has already navigated a path from A to C for example such as is shown in Figure 24A and now they are familiar with the topology of the virtual world wants to be automatically navigated from A to C but see more sights along the way. As the user is more familiar/better able to cope with the virtual environment (for example, a path may be more wiggly if an adult is exploring

than if a very young child is exploring), the user is able to retain their sense of orientation despite several changes in direction and a more wiggly path can be followed if the user assigns a relatively high LoS angle differential ranking for the navigational nodes along the guided path.

Such a constraint can also be implemented using the user profile described in more detail later below.

The heuristic is an estimate of the cost to get from the current node to the destination node, and one possible choice for this is:

heuristic = euclidian_distance_to_destination

The heuristic in the embodiment is therefore based on the Euclidian distance to destination (i.e. a straight line from the current node to the destination node), although it could alternatively also include some measure of the distance, interest and angle weightings w1, w2 and w3 if desired.

The system therefore runs through the A* algorithm for the nodes using the evaluation function, until the final destination node is reached. Importantly, the weightings w1, w2, and w3 can be varied depending on the user profile.

Defining a user profile

A suitable interface (not shown) allows the user to define their profile (e.g. what they are particularly interested in, and how they like to navigate through the virtual world). The user profile settings are then incorporated into the weightings above used for the A* algorithm, which consequently will result in different paths being generated for different users. The translation of the user interests into weightings allow different routes to be compared against each other such that the optimum path for that user can be generated. It can be seen that for a user who is particularly interested in visiting features of interest (paintings, landmarks, etc) it is likely to be the case that the system will not calculate the quickest or shortest route between two points. Rather, the lowest cost path (i.e. the best route for that user) will be one which visits as many features of interest as possible.

The user may input their profile upon first entering the virtual world. Some options include:

Always select shortest path	on / off
Always select quickest path	on / off
Ignore all features	on / off
Gaze orientation	on / off

Velocity control	on / off
Importance of path being short (w1)	weighting
Importance of path not being wiggly (w3)	weighting
Importance of visiting Feature Group 1 [e.g. paintings] (w2)	weighting
Importance of visiting Feature Group 2 [e.g. information points] (w2)	weighting
Importance of visiting Feature Group 3 [e.g. landmarks] (w2)	weighting

The first five options are overrides, and are thus non-weighted. For example, if the user selected "Always select shortest path" as ON, then the weighting w2 for the features of interest become irrelevant and will be set to zero.

The final five options are all weighted, and however the weightings are distributed they should all be normalised (e.g. add up to 100% in total). Any option which is not important to the user and not given a weighting is deemed irrelevant and not taken into account in the pathfinding process.

Dynamic Navigational Nodes

In addition to the fixed navigational nodes (29A, 29B, 29C, etc), defined initially by the content provider, the system may also include dynamic navigational nodes (not shown). These nodes operate like navigational nodes in the sense that they can form part of the path generated for the user, but provide enhanced functionality because the user is able to move them. A dynamic navigational node is associated with an interactive mesh object in the virtual world environment that can be moved around by the user. The user is able to specify that they want the dynamic navigational node to be a destination point, and the path will be calculated to it's location.

If a dynamic navigational node is placed a large distance from other navigational nodes, then the subdivision process described earlier may be used to fill in new nodes until a suitable node density is achieved in that area. This then allows a smooth path to be calculated to the node.

If a dynamic navigational node is placed without a line of sight to any other navigational nodes (i.e. in a blind spot) then the node is defined as invalid – lost from the Node Matrix (although still present in the environment) and the user is prevented from selecting it as a destination. Alternatively, the system may prevent the user from placing the node in a blind spot, and only allows it to be moved into valid positions.

Switch Navigational Nodes

In addition to the fixed navigational nodes and the dynamic navigational nodes, the system may also include switch navigational nodes (not shown), defined by the content producer. As mentioned earlier, navigational nodes have a Level ID, specifying which level the node belongs

to, and effectively grouping the nodes into different subsets for more efficient path calculation. For example, if the virtual world consists of two different rooms, then the navigational nodes in one room may be defined as belonging to one level, and the navigational nodes in the second room to a different level. During the pathfinding process in one room, only the nodes of that level are evaluated, saving takes into account nodes on a different level. The switch nodes then provide the functionality to swap from one level to the second level (e.g. the switch navigational node might be positioned in a doorway between the two rooms).

Associated with each switch navigational node is the following data set:

Node ID
Node Level ID #1
Node Level ID #2
XYZ position
LoS data
Switching data

Node ID – an identifier for the node
Node Level ID #1 – the identifier for the first level
Node Level ID #2 – the identifier for the second level
XYZ position– these are the x, y, z co-ordinates giving the location of the switch navigational node
LoS data – this gives the data for all nodes (for both levels) which the present node has Line of Sight (LoS) to
Switching data – gives the criteria of when the generated path can / cannot switch from navigational nodes on one level to nodes on the other level.

Event Feature Nodes

As described earlier, the content provider is able to define feature nodes (28A, 28B, 28C, etc) associated with the features of interest in a virtual world. The content producer is also able to define event feature nodes (not shown) which have the extra functionality that they react to users within the virtual world, or are only active at certain predefined times.

These event feature nodes expand the functionality of the system into the gaming and interactive online environment platforms, where multiple users can interact with the environment and with each other (e.g. the Event Feature may simulate an explosion, or the opening and closing of an interactive chat-room / race event. The content provider must specify when the event feature will be active / inactive, or what triggers it will respond to. An example data set is as follows:

Node ID
Feature Group ID
Trigger data
XYZ of pivot point data
Sphere of influence settings
Time settings
Display settings

Node ID – an identifier for the node (e.g. text string "explosion")

Feature Group ID – an identifier for the type of feature associated with the node

Trigger data – the conditions that will cause the event feature node to become active

XYZ of Pivot Point Data – these are the co-ordinates giving the location of the event feature node

Sphere of influence settings – these define (for a 3D world) a three-dimensional sphere surrounding the feature node. However, in addition to the settings used for the standard feature nodes, these additionally include a time dependant feature that may allow the sphere of influence to degrade over time.

Time settings – these define when the event feature node will be active / inactive

Display settings – these are the settings to be used for text / images associated with the feature when constrained

Further functionality

As explained above, the processing and pathfinding stages result in a spline path that will be used to take the user through the environment. The spline path therefore comprises a plurality of nodes each joined by links to the next node. The system applies smoothing to this path by interpolating the spline so as to improve further the user experience.

During real time display, when the user is guided through the environment they have the further option to control their field of view. As discussed earlier, the content provider can specify particular angles of viewing (FoV constraints) which are to be applied for specific feature nodes so as to ensure the user does not miss features of interest. The user can choose to toggle control to either operate this automatic field of view presentation or alternatively to control

manually their own field of view.

In the embodiments described above, it is explained that the processing stage which creates the more dense matrix of nodes occurs after the content provider has defined the contents within a virtual world, but before the system is used by a user desiring to navigate within that world. It is of course alternatively possible for either part or all of the processing stage which adds these subdivided nodes to be carried out in real time whilst the system is in use by a user. Indeed, an alternative arrangement is particularly desirable when the virtual world contains a number of dynamically moving objects (for example, a virtual environment of a city might include moving cars, buses, etc). In this dynamic case, only some subdivision (for the static objects such as buildings, etc) could be carried out in advance before a user accesses the system, with the remaining processing to add the extra nodes occurring each time a user specified that they desire a path to be created. The processing stage thus takes into account the positions of these dynamic objects at the instant the user desires the path to be created. This functionality could be supplemented by defining some regions in the world in which this subdivision will occur (for example, streets within a virtual city), and other regions in the world when the dynamic subdivision will not occur because they contain only stationary objects (for example, the insides of some buildings). This is a typical example when switch nodes (mentioned above) are used to delimit the different regions.

Furthermore, it may be necessary in real time to create new nodes corresponding to the start and destination points selected by a user for a path generation procedure. If nodes do not already exist at these locations then the system may be used to automatically create such nodes, and additionally use processing to add new subdivided nodes in only the regions close to the start and destination nodes so as to link them to the already existing node matrix.

This invention enables the overall ranking for each prospective node path from the start to the destination points of a guided path to incorporate a rating based on the LoS Angle Differential. The LoS Angle Differential rating is first normalised, however, to enable the weighting indicated in a User Profile to influence the path ratings. After the set values within a VW have been weighted against the user settings the actual path rating calculation is performed.

The LoS Angle Differential also enables the content producer of a VW to manually define constraints to the LoS angle between nodes to crop the angles which are searched for node partners without the need for a collisionable mesh, such as is shown in Figure 25A.

Figure 25A shows a navigational node for which a valid LoS angle to or from another node is constrained to a range of angles between 0° and θ°, where $\theta < 360°$ ($\theta° = 360°$ would remove the range constraint for the LoS angle between nodes.

This functionality allows the creation of channels of nodes that ignore other nodes around them which produces a break in the node matrix where links are not desiredFigure 25B shows how this may modify the LoS between two navigational nodes A and D, so that instead of a path being generated along a LoS connection between A and D, the pathfinding system instead generates a path via navigational nodes B and C. Navigational node B is positioned within the range of exit angles for navigational node A, whereas navigational node C is positioned between the range of entry angles for influence of node D.

It is possible for a node to have both a range of entry angles (and "entry cone") and a range of exit angles (an "exit cone")such as Figure 26 shows for node B. . An additional LoS constraint allows the content producer to create a first LoS "cone" which defines a range of angles that the first LoS line to the previous node on the path must enter through, and a second LoS "cone" which defines a range of angles through which a second LoS to the next node along the path must exit the node.

In Figure 26, the pathfinding system is generating a path between nodes A and C, and node B is constrained in that it can form connections only with nodes within its entry and exit cones. Thus in Figure 26, the LoS between node A and node B falls within the entry cone for node B. Similarly the LoS between node B and node C falls within the exit cone for node B. However, the LoS between node D and node B and the LoS between node B and node E lie outside the range of permitted approach and exit angles for node B. Accordingly, the dashed lines between nodes D and B and B and E are invalid connections.

The LoS constraint shown in Figure 26, can be implemented as a modified version of the LoS Angle Differential constraint/weighting system described herein above with reference to Figures 23A to 23C and 15A and 24B of the accompanying drawings. For example, the range of angles of a second LoS cone (exit cone) is determined as a function of the LoS Angle Differential and the range of angles of a first LoS cone (entry cone), or vice versa in some embodiments of the invention.

Whilst the above embodiment of the invention has been described in the context of a pathfinding system suitable for plotting a guided path along which a user is automatically navigated in a virtual environment, the implementation of applying a weight to a pathfinding system which is based on the extent to which the path bends can be used in other systems in which a user wishes to plot a path, for example systems which enable the user to navigate across land and/or sea environments. In particular, where a user is implementing a system using global positioning satellite technology or the like, especially where the environment enables the user can determine a path which follows a relatively unconstrained route, the

invention can be used to provide the user with a scenic route (for example, one which meanders across a desert or island-hops in a sea-environment) or a direct route (which might be more efficient in terms of fuel economy).

Those skilled in the art will appreciate that the scope of the invention is determined by the appended claims, and that where a feature known to those skilled in the art is an apparent equivalent of a feature described herein, a feature described herein should be interpreted as synecdoche for all equivalent features. The features described herein with reference to particular embodiments may be combined with any particular aspect and with other features in any appropriate manner apparent to those skilled in the art.

CLAIMS

1. A method of generating a new path to a destination node in a virtual environment comprising a plurality of nodes, the method comprising:

storing nodal information identifying one or more nodes associated with a previously created path to said destination node;

dynamically reconfiguring the topology of the virtual environment to define a start node for said new path; and

processing said stored nodal information to determine the new path to said destination by including at least one node of said previously created path.

2. A method as claimed in claim 1, wherein when said previous path is no longer actively navigated along, an exit point is determined for said previous path.

3. A method as claimed in claim 2, wherein said exit point comprises a node on said previous path.

4. A method as claimed in claim 2, wherein said virtual environment is dynamically changed to include an node on said previous path, if said exit point does not comprise a node on said previous path.

5. A method as claimed in any previous claim, wherein in said step of processing, a return path to said previously created path which intersects said previously created path at the nearest node to said start node for said new path is determined, and said new path includes said return path.

6. A method as claimed in claim 5, wherein said nearest node is determined by processing in parallel return paths to all nodes on said previous path.

7. A method as claimed in any preceding claim dependent on claim 2, wherein said new path includes all subsequent nodes of said previous path between said exit point and said destination.

8. A method as claimed in any preceding claim dependent on claim 2, wherein in said step of processing, a return path to said previously created path which intersects said previously created path at the exit point is determined, and said new path includes said return path.

9. A method as claimed in any previous claim, wherein in said step of processing said stored nodal information to determine the new path to said destination by including at least one node of said previously created path, said nodal information is processed to increase the Interest_component of the nodes of said previous path.

10. A method as claimed in any previous claim, wherein the navigation mode employed to navigate along said previously generated path is the same as the navigation mode employed to navigate along said new path.

11 A method as claimed in claim 10, wherein a user is guided along said previous path in an automatic navigation mode, and wherein said user is able to switch to a manual navigation mode to create a manually navigated path to said start node of said new path, and wherein said user is guided along said new path to said destination in an automatic navigation mode.

12. A method as claimed in claim 11, wherein said participant is able to switch between automatic navigation modes and manual navigation modes of their own volition at any point in said virtual environment.

13. A method as claimed in claim 10, wherein, when said participant switches from said automatic navigation mode to said manual navigation mode, an exit node is created along said previously created path.

14. A method of generating a return path to a portion of a previously created path in a virtual environment, wherein a user is able to be automatically navigated along the previously created path from a start node to a destination node, the virtual environment supporting the ability of the user to switch at their own volition from an automatic navigation mode along said previously created path to a manual navigation mode generating a second path which deviates from said previously created path in said virtual environment, the method comprising:

storing information identifying said start node, destination node and any intermediate nodes of said previously created path in said virtual environment;

dynamically reconfiguring the topology of the virtual environment to define a second start node at the point where said second path terminates; and

processing said nodal stored information to determine a return path from said second start node which intersects said previously created path at a re-entry node.

15. A method as claimed in claim 14, wherein an exit node is defined at a point along said first path where said second path commences, and said step of storing information comprises storing information identifying said exit node, start node, destination node and any intermediate nodes of said first path in said virtual environment.

16. A method as claimed in claim 14 or 15, further comprising performing the step of validating said first path using said stored information, and if said first path is still valid, performing the step of:

processing said nodal stored information to determine a return path from said second start node which intersects said first path at a re-entry node.

17. A method as claimed in any one of claims 14 to 16, further comprising automatically navigating the user along the return path and the continued portion of said first path to said destination.

18. A method as claimed in any one of claims 14 to 17, wherein in said step of processing said stored information, a parallel node search is performed from said second start node for all saved nodes of said first path to determine a return path from said second start node to a saved node of said first path.

19. A method as claimed in any one of claims 14 to 18, wherein the closest saved node of the first path is determined as said re-entry node.

20. A method as claimed in any claim dependent on claim 15, wherein the exit node of the first path is determined as said re-entry node.

22. A method as claimed in any previous claim, wherein said user is a participant in said virtual environment.

23. A system arranged to support the generation of a new path to a destination node in a virtual environment comprising a plurality of nodes, the system comprising:

means to store nodal information identifying one or more nodes associated with a previously created path to said destination node;

means to dynamically reconfigure the topology of the virtual environment to define a start node for said new path; and

processing means to process said stored nodal information to determine the new path to said destination by including at least one node of said previously created path.

24. A system arranged to support the generation of a return path to a portion of a previously created path in a virtual environment, wherein a user is able to be automatically navigated along the previously created path from a start node to a destination node, the virtual environment supporting the ability of the user to switch at their own volition from an automatic navigation mode along said previously created path to a manual navigation mode generating a

second path which deviates from said previously created path in said virtual environment, the system comprising:

means to store information identifying said start node, destination node and any intermediate nodes of said previously created path in said virtual environment;

means to dynamically reconfigure the topology of the virtual environment to define a second start node at the point where said second path terminates; and

processing means to process said nodal stored information to determine a new path including return path from said second start node which intersects said previously created path at a re-entry node.

25. A system as claimed in either claim 23 or 24, wherein the system further comprises means to enable a user to be automatically navigated along said previously created path.

26. A system as claimed in claim 26, wherein the system further comprises means to enable a user to switch from an automatic navigation model along said previously created path to a manually navigation mode and back to an automatic navigation along said new path.

27. A user interface for a computer system as claimed in claim 26, wherein the user interface is arranged to enable a user to switch from an automatic navigation model along said previously created path to a manually navigation mode and back to an automatic navigation along said new path.

28. A computer program comprising instructions for causing one or more processors to perform the method according to any of claims 1 to 22 when the instructions are executed by the processor or processors.

29. A computer data signal embodied in a carrier wave and representing instructions for causing one or more processors to perform the method according to any of claims 1 to 22 when the instructions are executed by the processor or processors.

30. A storage medium carrying computer readable code representing instructions for causing one or more processors to operate as the system according to any of claims 23 to 26 when the instructions are executed by the processor or processors.

31. A computer program comprising instructions for causing one or more processors to operate as the system according to any of claims 23 to 26 when the instructions are executed by the processor or processors.

32. A computer data signal embodied in a carrier wave and representing instructions for causing one or more processors to operate as the system according to any of claims 23 to 26 when the instructions are executed by the processor or processors.

33. A storage medium carrying computer readable code representing instructions for causing one or more processors to perform the method according to any of claims 1 to 22 when the instructions are executed by the processor or processors.

ABSTRACT

PATHFINDING SYSTEM

A method of generating a new path to a destination node in a virtual environment comprising a plurality of nodes. the method comprises storing nodal information identifying one or more nodes associated with a previously created path to said destination node, dynamically reconfiguring the topology of the virtual environment to define a start node for said new path, and processing said stored nodal information to determine the new path to said destination by including at least one node of said previously created path.

Figure (14)

Figure 1

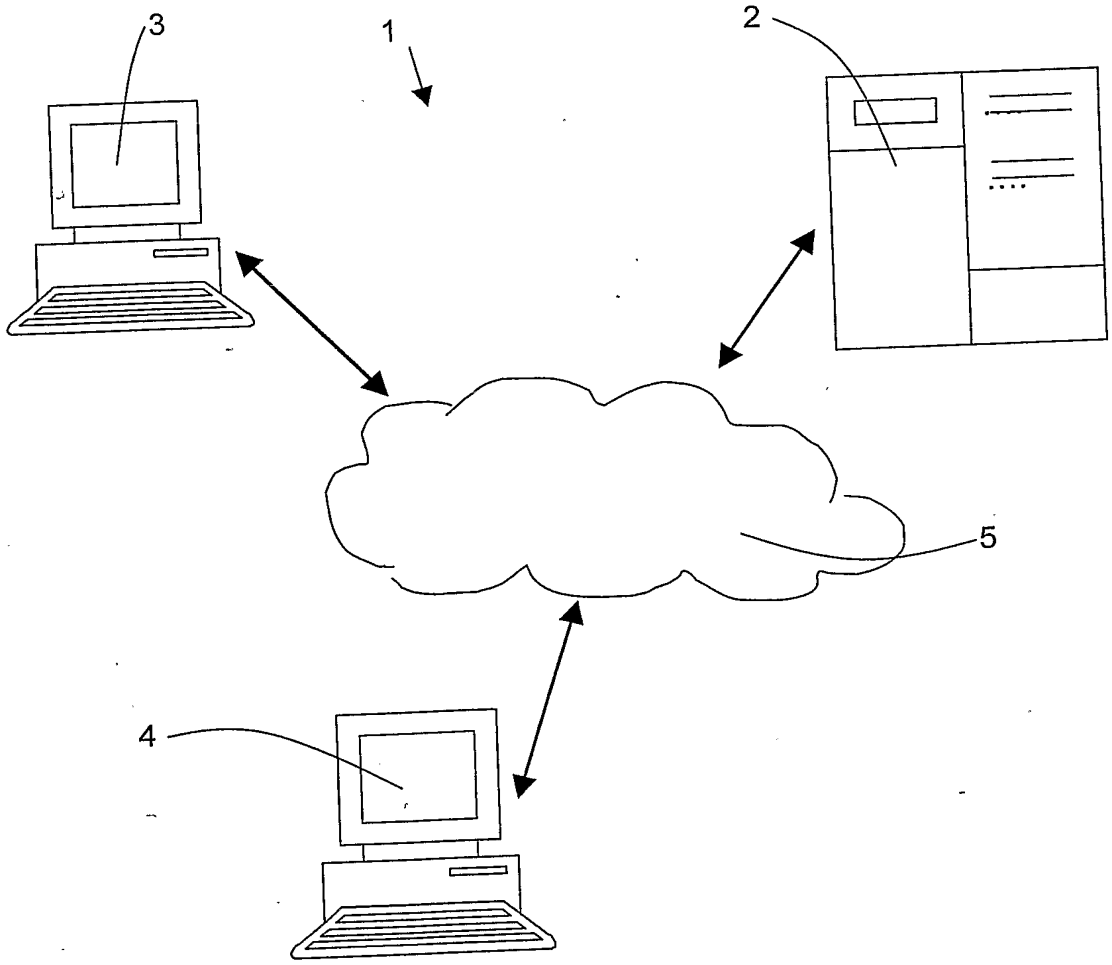


Figure 2

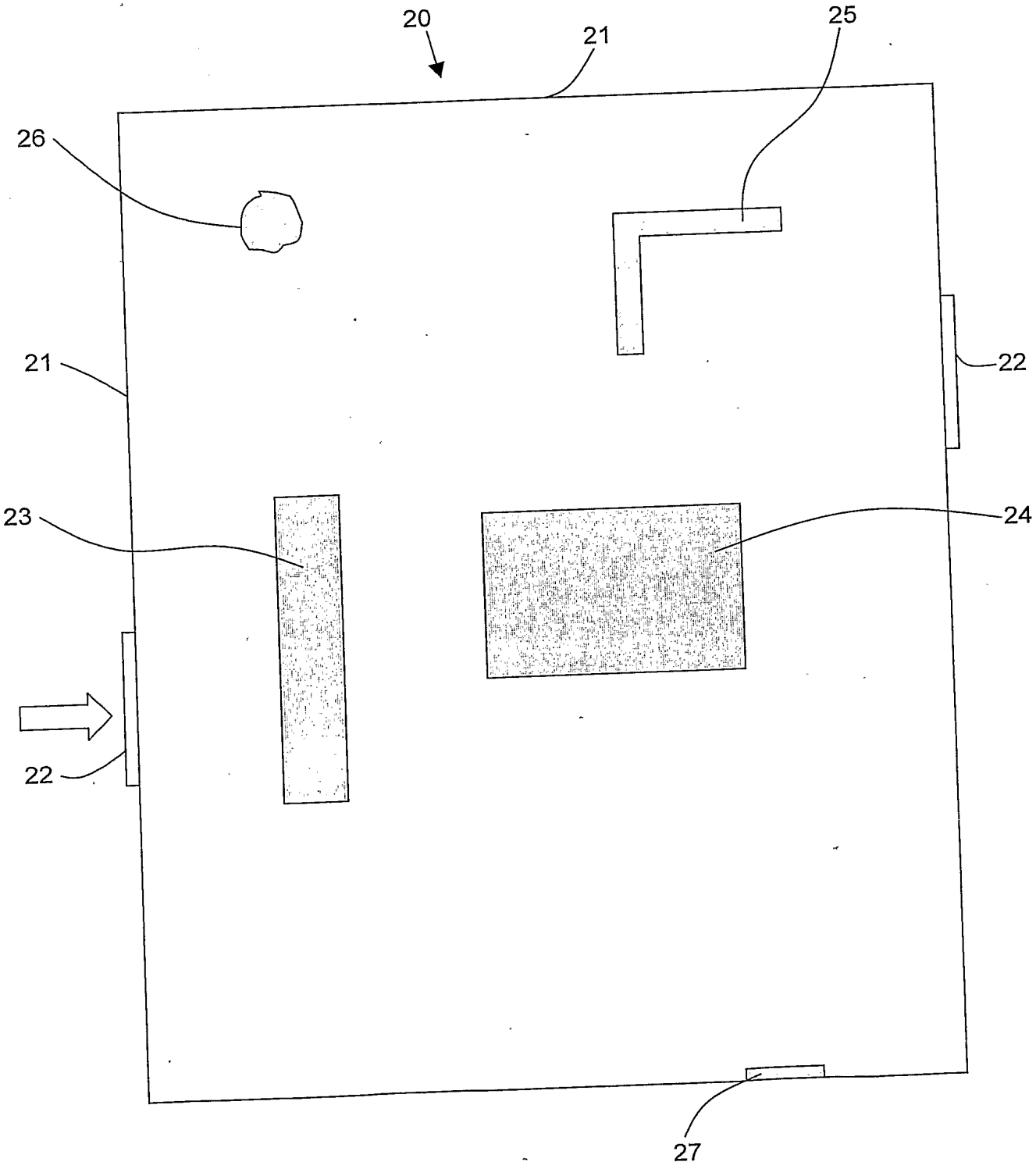


Figure 3

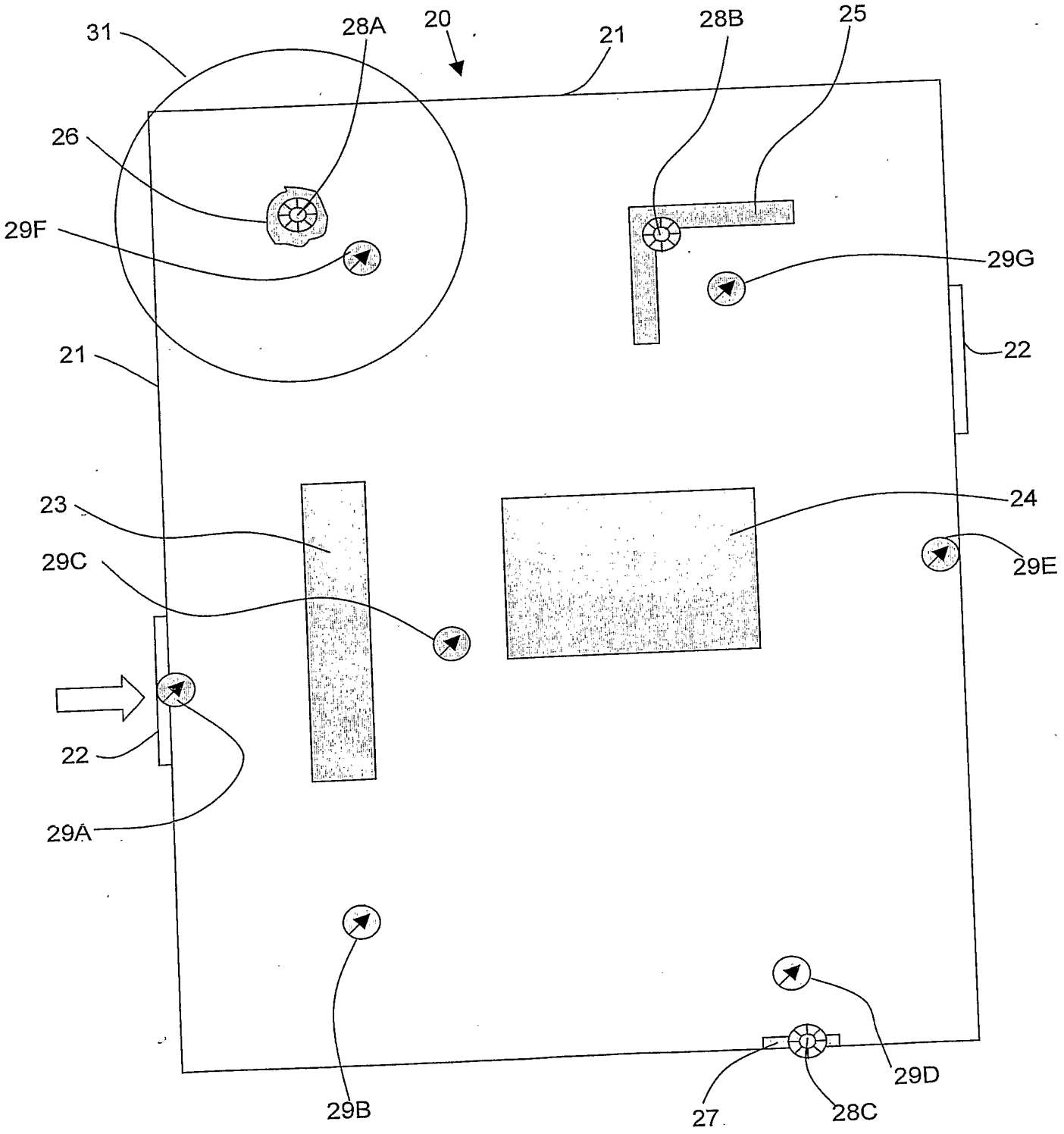

31
28A
20
21
28B
25
26
29F
29G
21
22
23
24
29C
29E
22
29A
29B
29D
27
28C

Figure 4

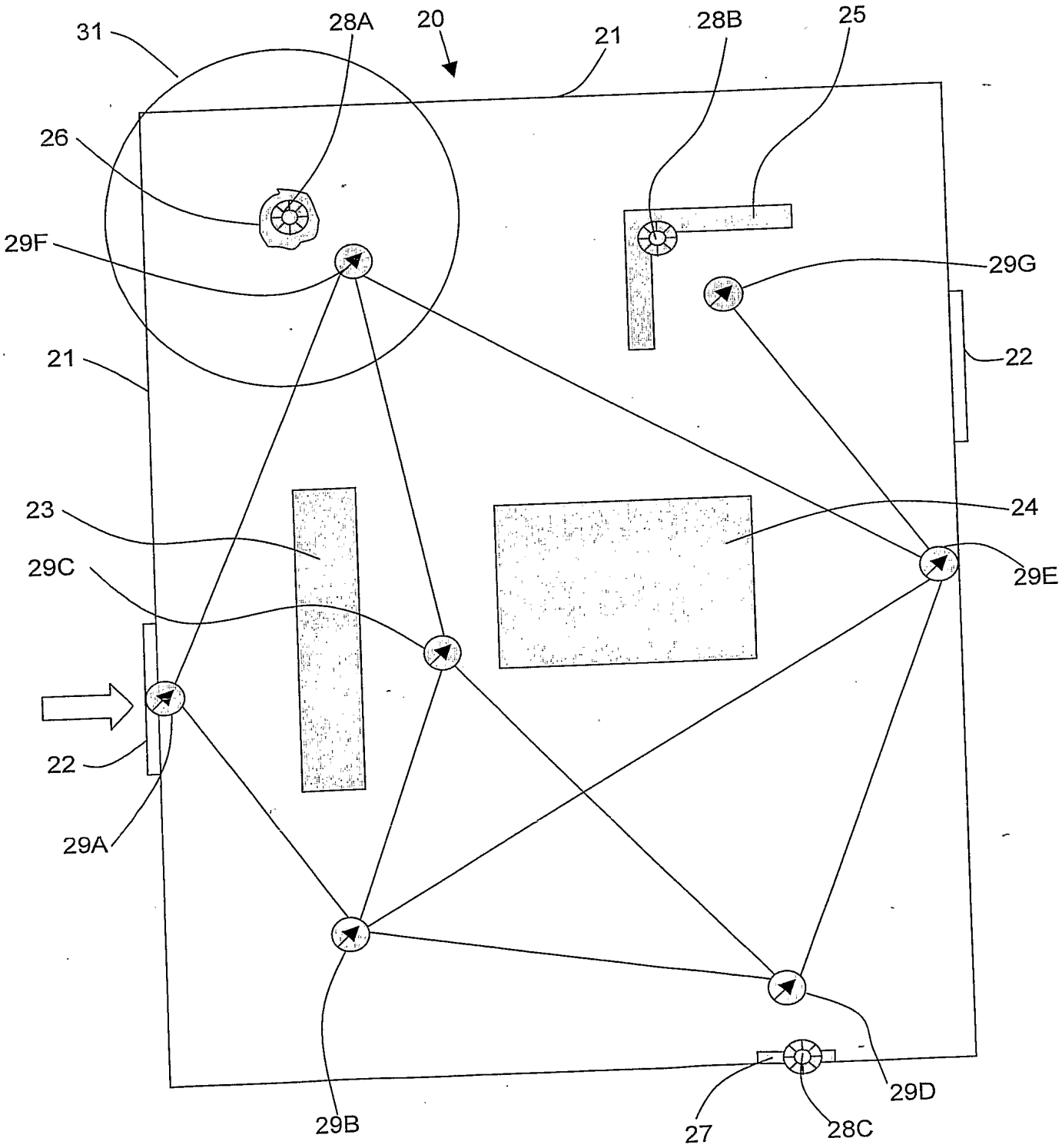

31
28A
20
21
28B
25
26
29F
29G
21
22
23
24
29C
29E
22
29A
29B
27
28C
29D

Figure 5

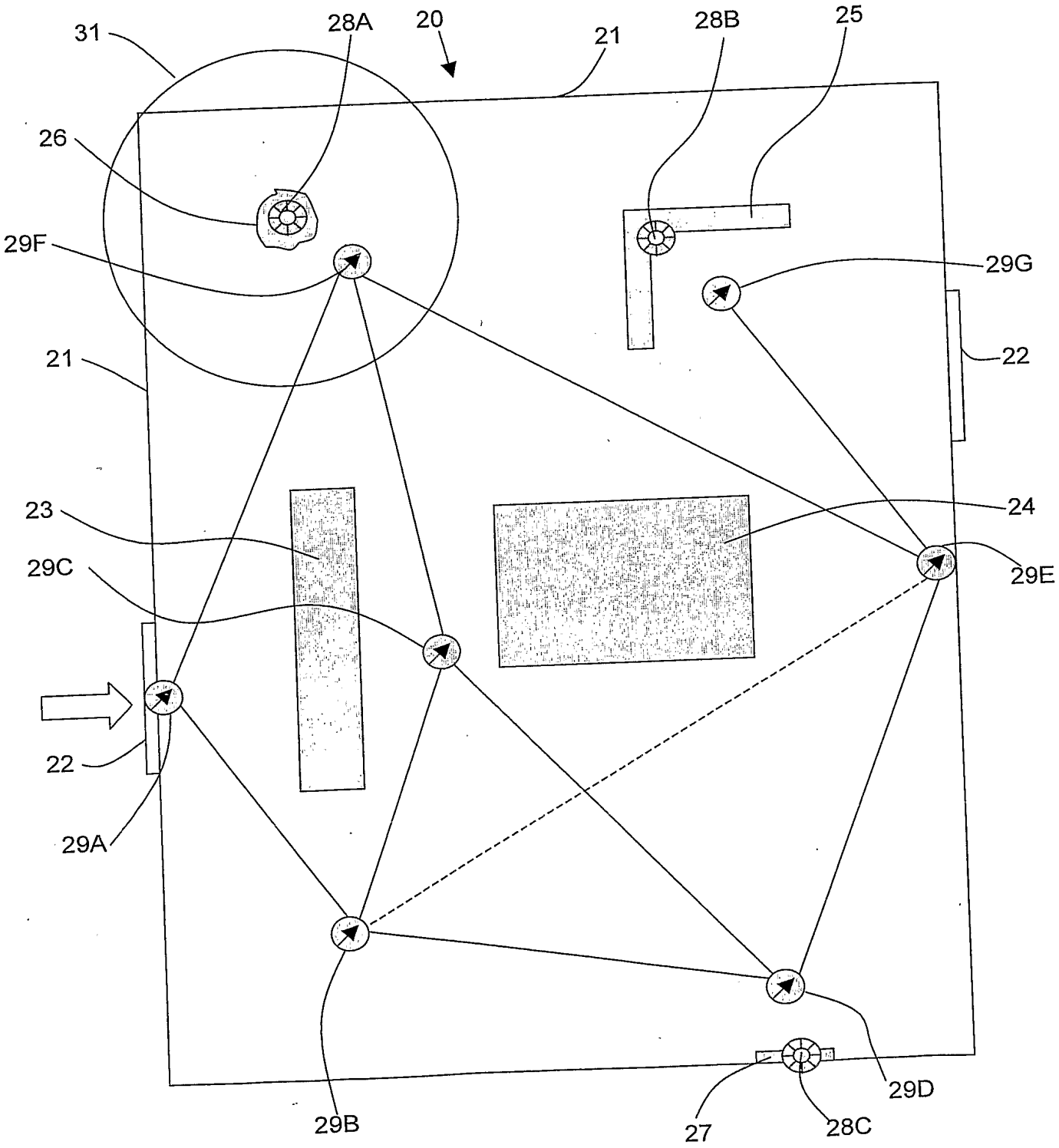

31
28A
20
21
28B
25
26
29F
29G
21
22
23
24
29C
29E
22
29A
29D
29B
27
28C

Figure 6

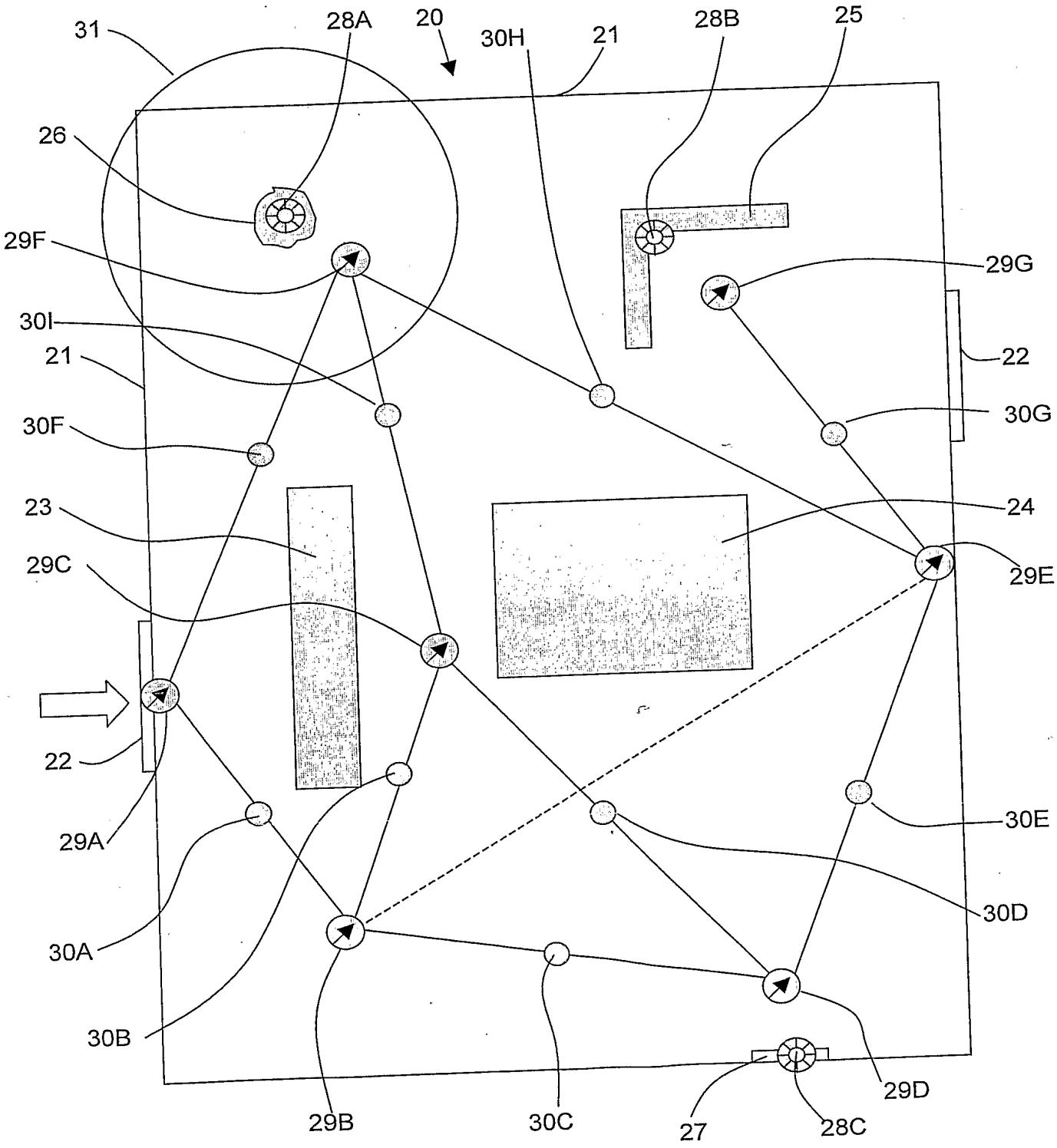

31
28A
20
30H
21
28B
25
26
29F
29G
30I
21
22
30F
30G
23
24
29C
29E
22
29A
30E
30D
30A
30B
29B
30C
27
28C
29D

Figure 7

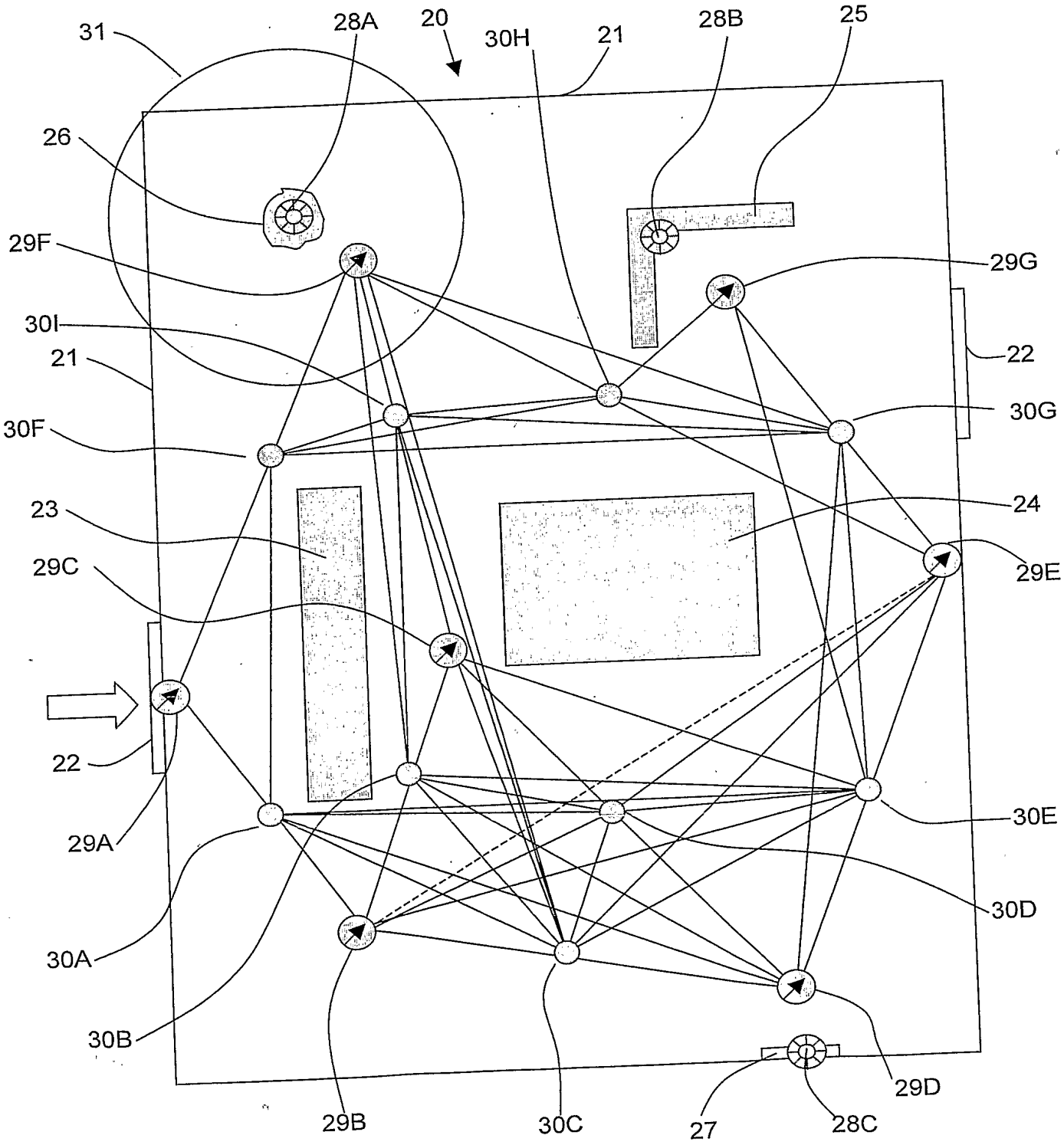

31
28A
20
30H
21
28B
25
26
29F
29G
30I
21
22
30F
30G
23
24
29C
29E
22
30E
29A
30D
30A
30B
29D
29B
30C
27
28C

Figure 8

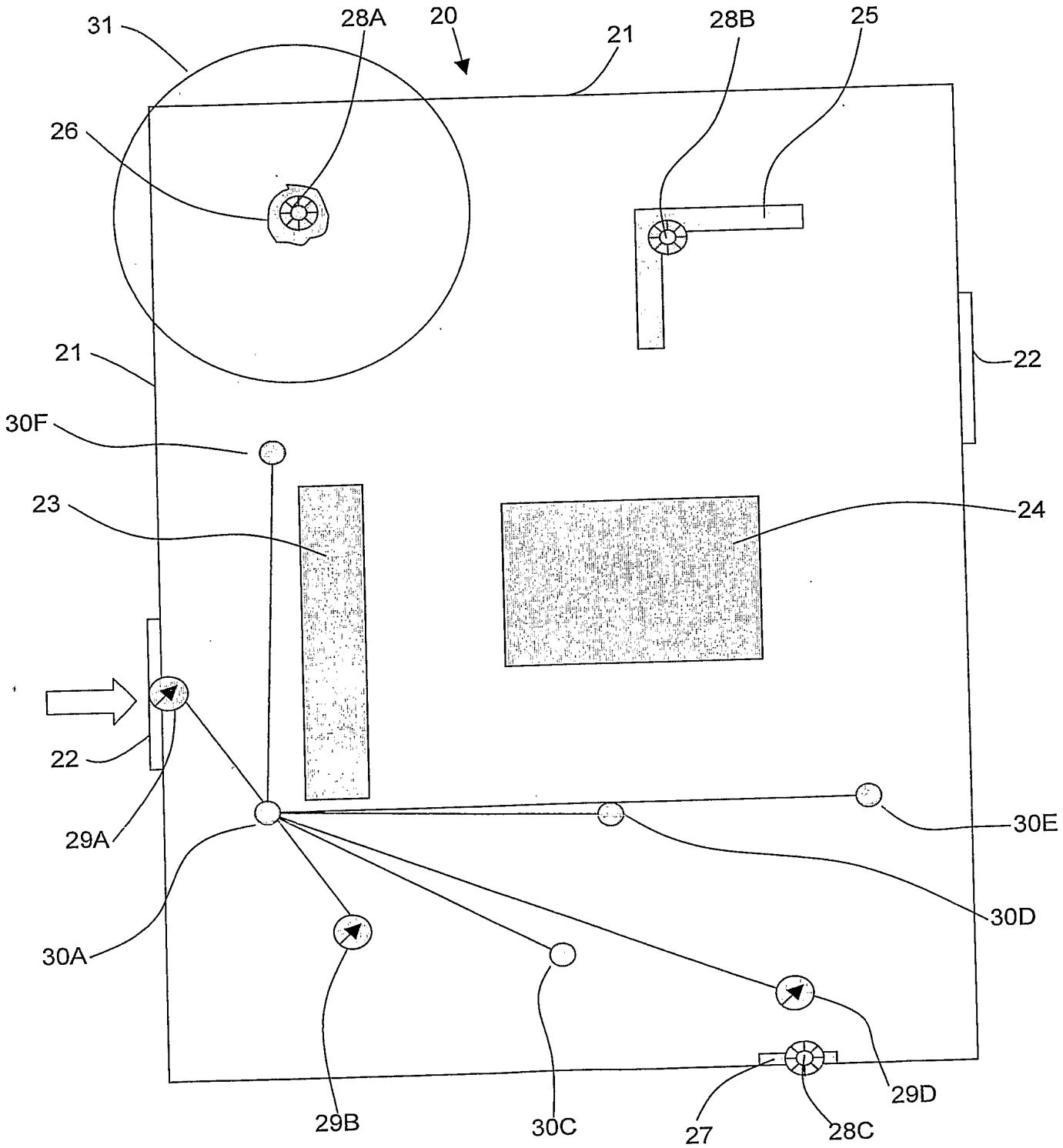

31
28A
20
21
28B
25
26
21
22
30F
23
24
22
29A
30E
30D
30A
29D
29B
30C
27
28C

Figure 9

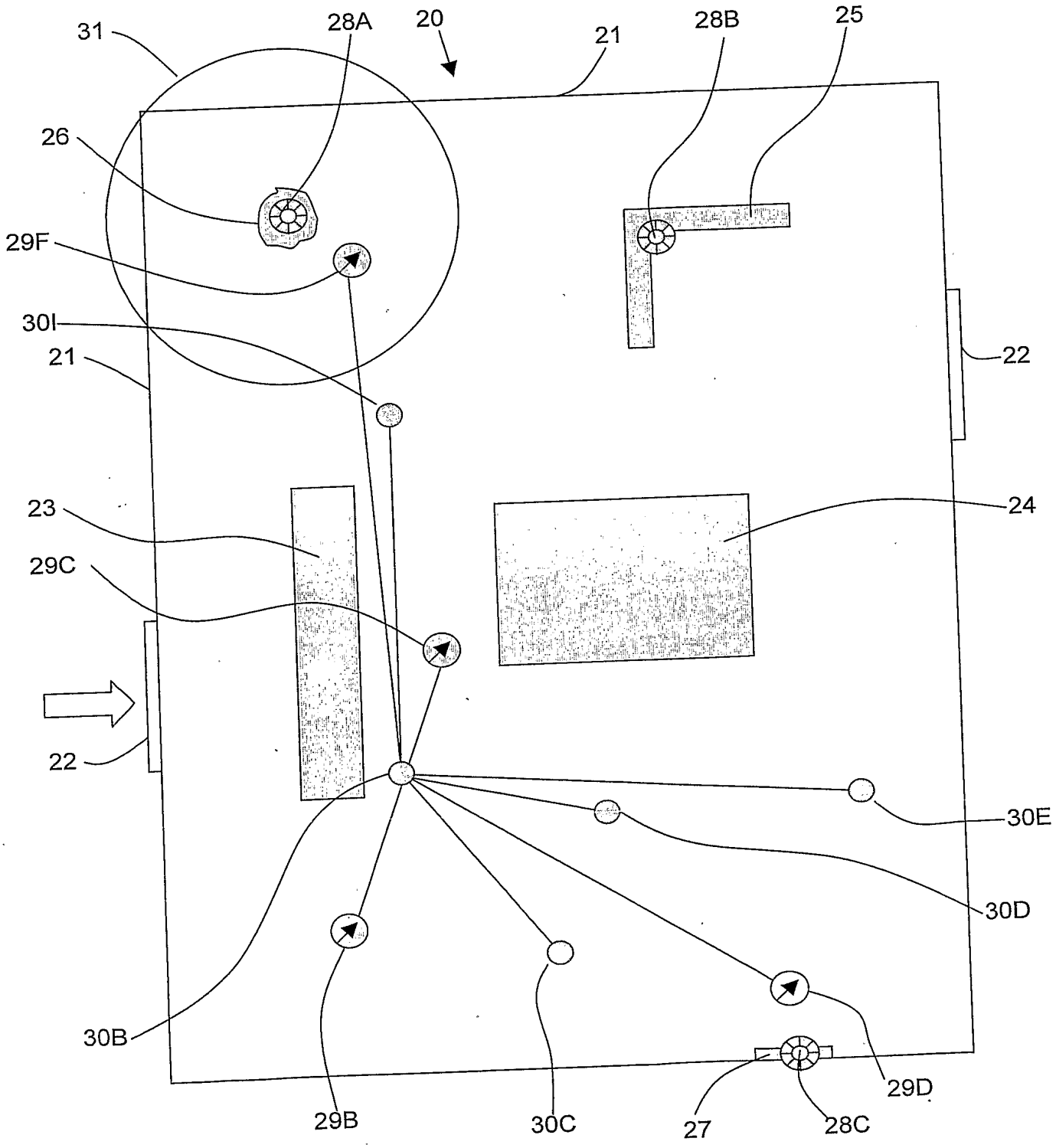

31
28A
20
21
28B
25
26
29F
30I
21
22
23
24
29C
22
30E
30D
30B
29D
29B
30C
27
28C

Figure 10

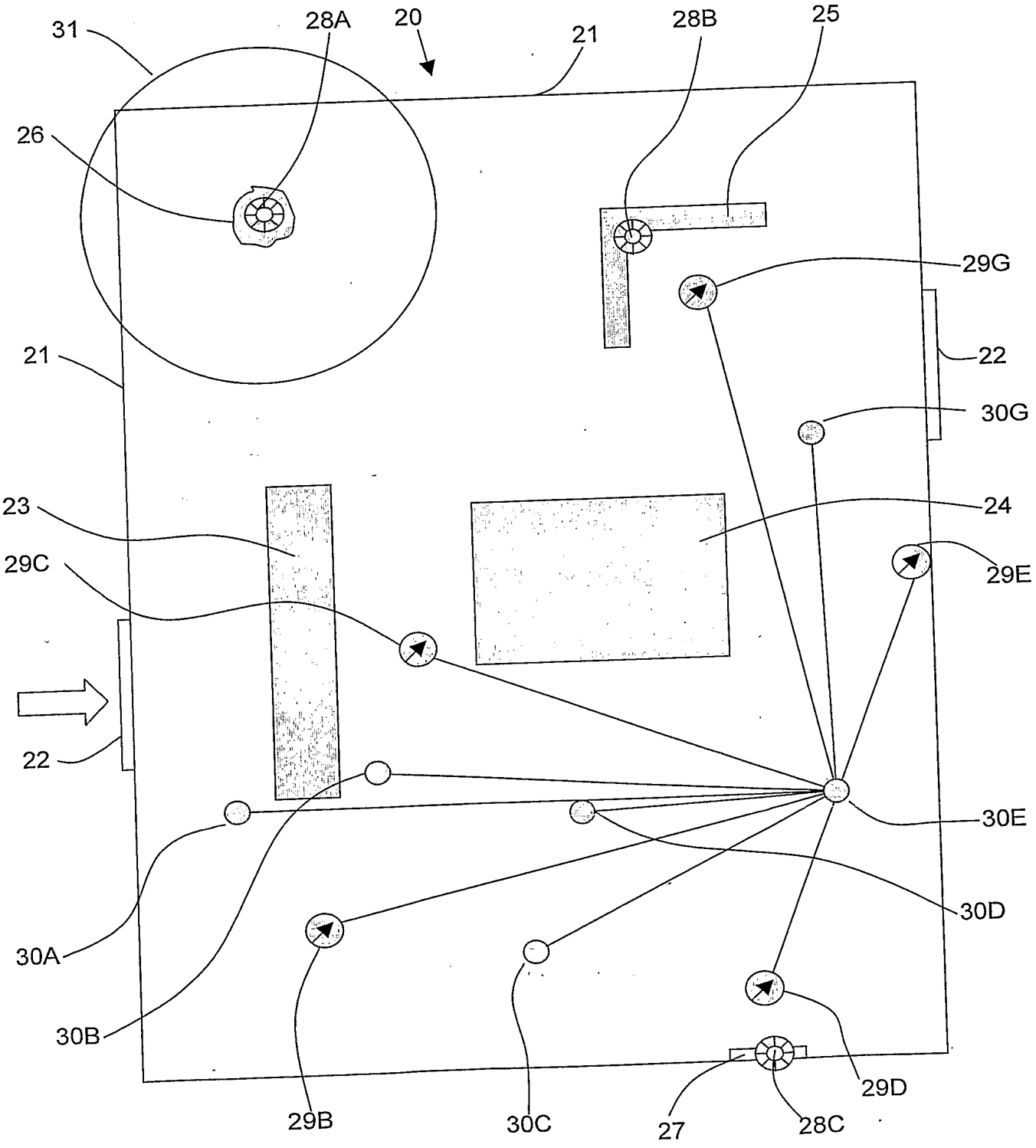

31
28A
20
21
28B
25
26
29G
21
22
30G
23
24
29C
29E
22
30E
30D
30A
30B
29D
29B
30C
27
28C

Figure 11

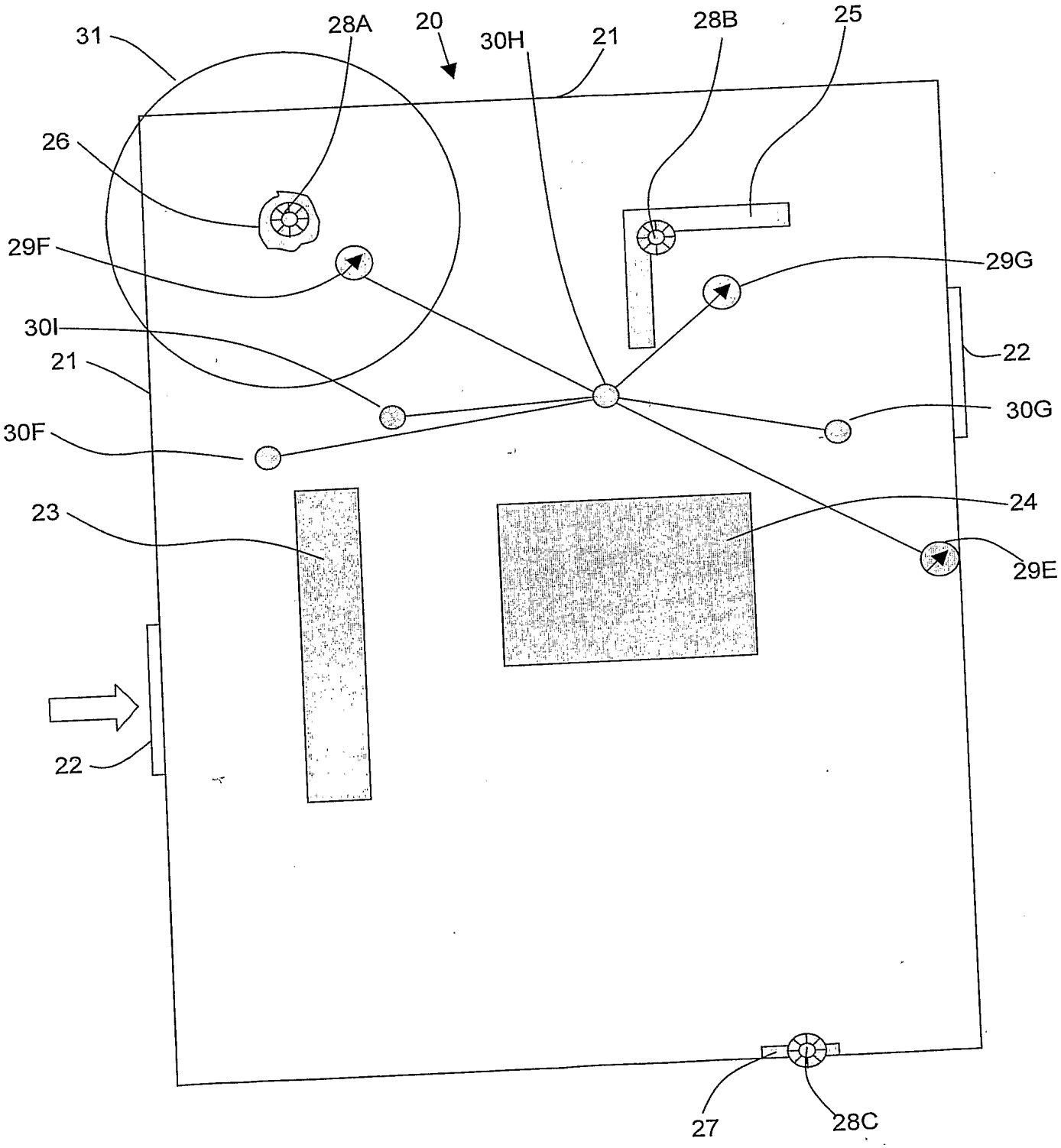

31
28A
20
30H
21
28B
25
26
29F
29G
30I
21
22
30G
30F
24
23
29E
22
27
28C

Figure 12

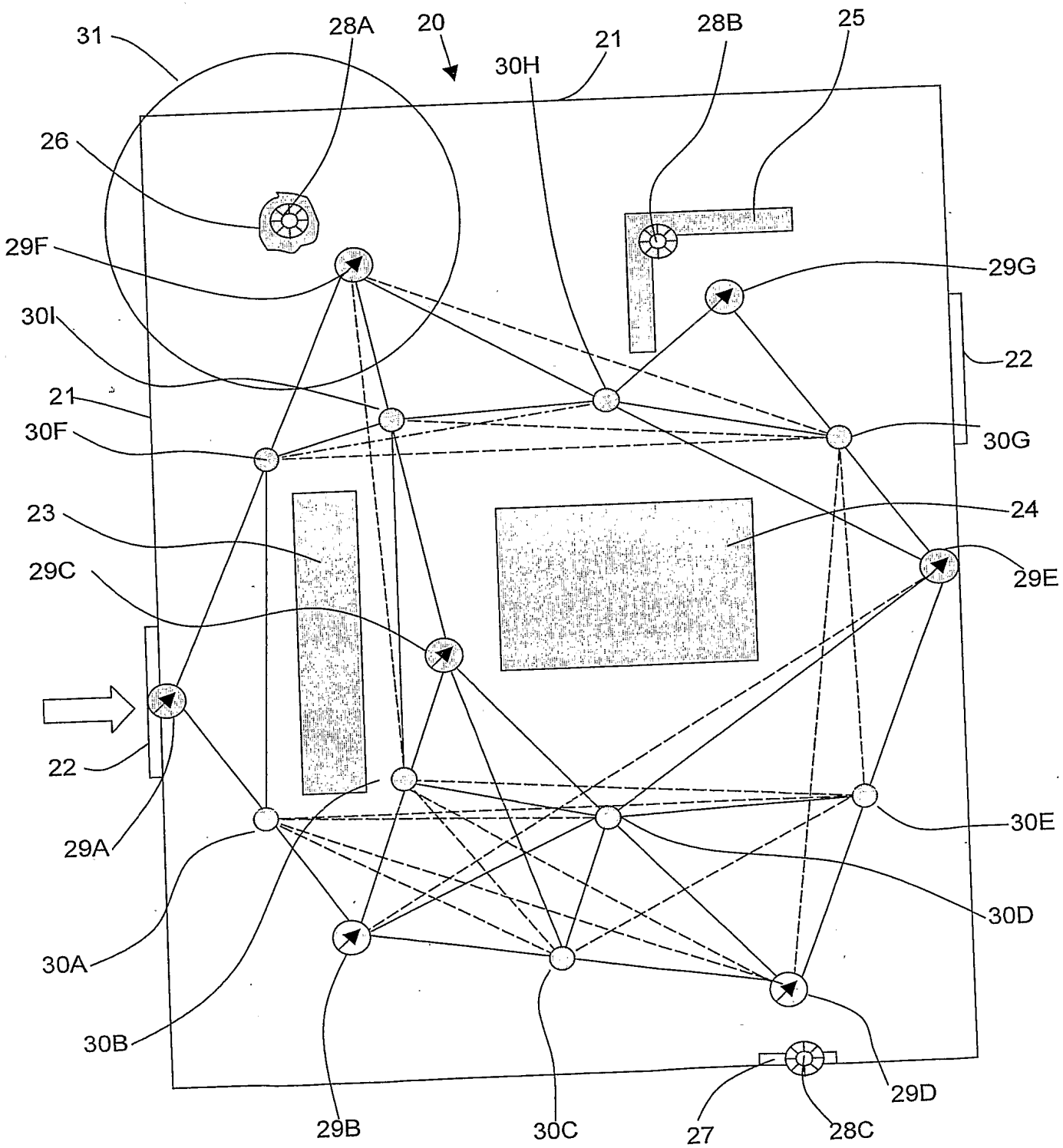

31
28A
20
21
28B
25
30H
26
29F
29G
30I
22
21
30F
30G
23
24
29C
29E
22
29A
30E
30D
30A
30B
29D
29B
30C
27
28C

Figure 13

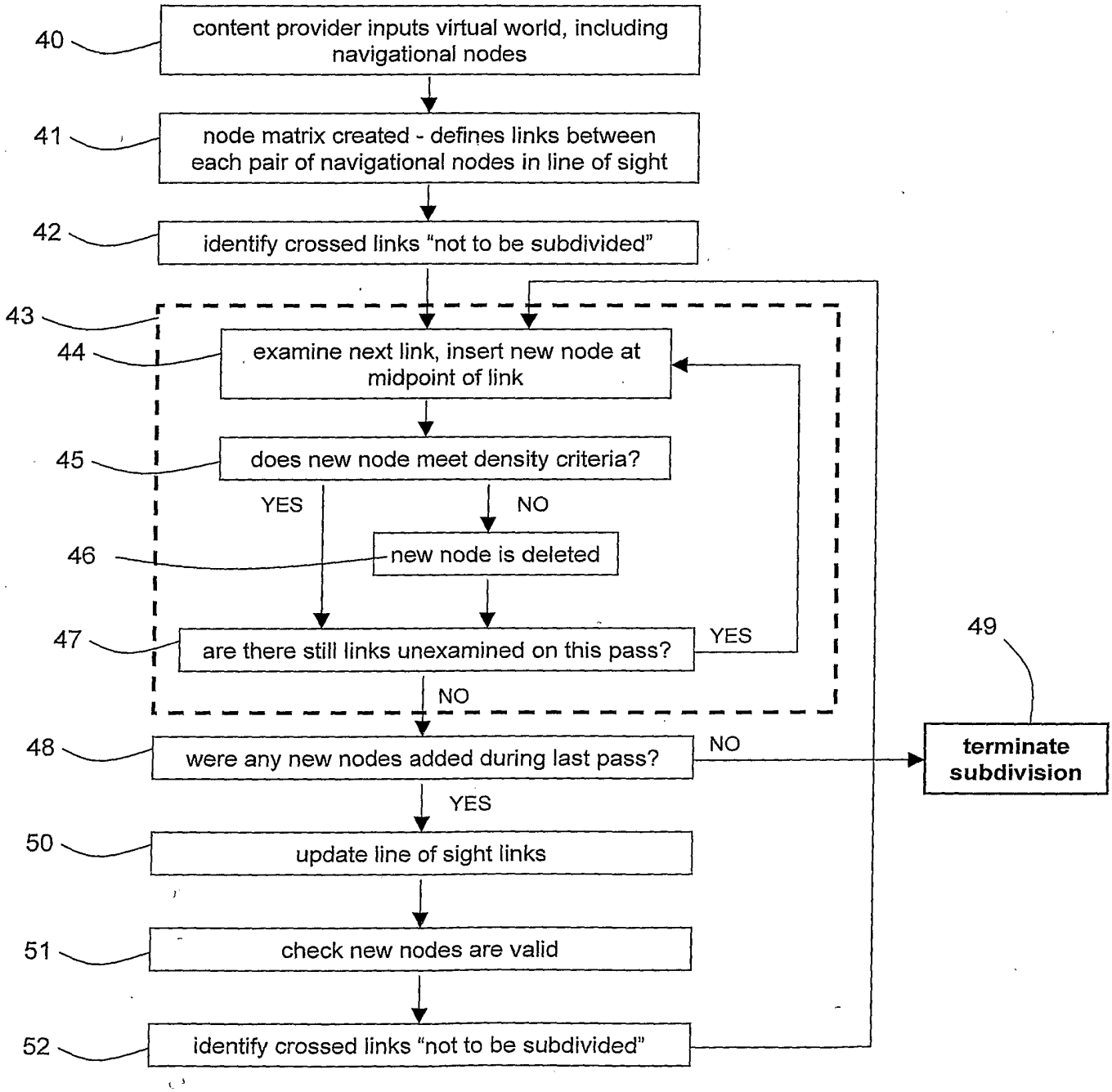

40
content provider inputs virtual world, including navigational nodes
41
node matrix created - defines links between each pair of navigational nodes in line of sight
42
identify crossed links "not to be subdivided"
43
44
examine next link, insert new node at midpoint of link
45
does new node meet density criteria?
YES
NO
46
new node is deleted
47
are there still links unexamined on this pass?
YES
NO
48
were any new nodes added during last pass?
NO
49
terminate subdivision
YES
50
update line of sight links
51
check new nodes are valid
52
identify crossed links "not to be subdivided"

Figure 14

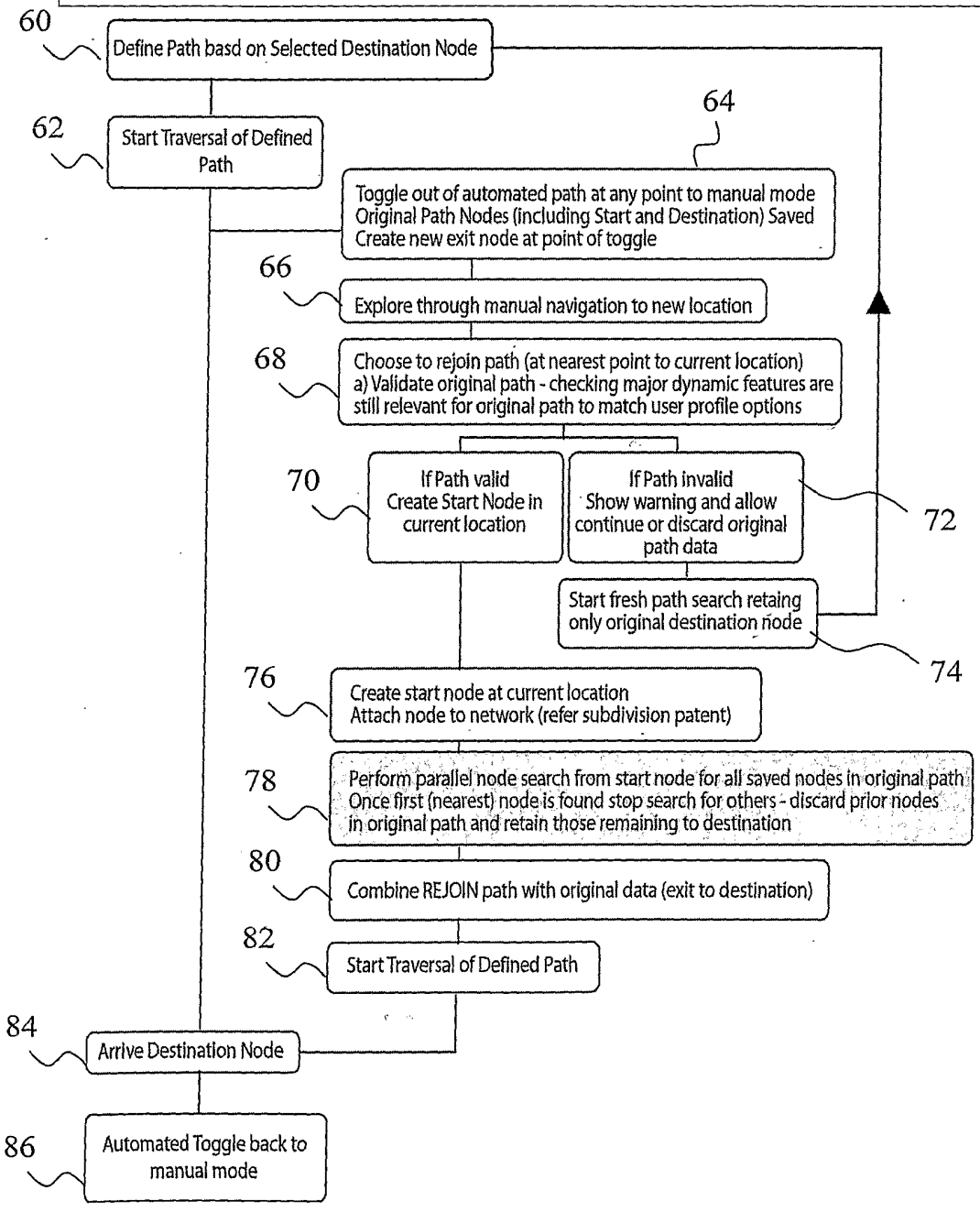

60
Define Path basd on Selected Destination Node
62
Start Traversal of Defined Path
64
Toggle out of automated path at any point to manual mode
Original Path Nodes (including Start and Destination) Saved
Create new exit node at point of toggle
66
Explore through manual navigation to new location
68
Choose to rejoin path (at nearest point to current location)
a) Validate original path - checking major dynamic features are still relevant for original path to match user profile options
70
If Path valid
Create Start Node in current location
If Path invalid
Show warning and allow continue or discard original path data
72
Start fresh path search retaing only original destination node
74
76
Create start node at current location
Attach node to network (refer subdivision patent)
78
Perform parallel node search from start node for all saved nodes in original path
Once first (nearest) node is found stop search for others - discard prior nodes in original path and retain those remaining to destination
80
Combine REJOIN path with original data (exit to destination)
82
Start Traversal of Defined Path
84
Arrive Destination Node
86
Automated Toggle back to manual mode

Figure 15

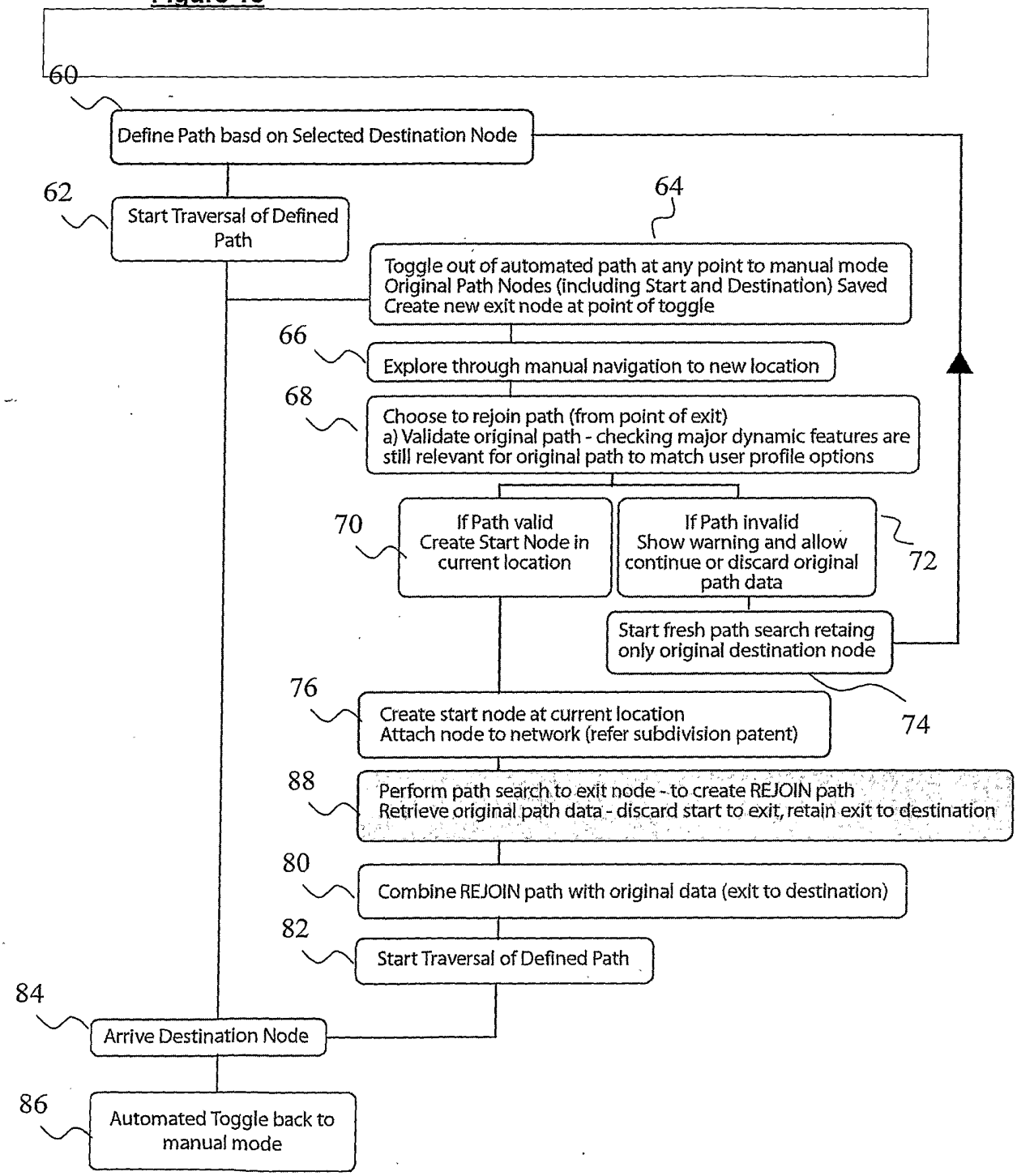

60
Define Path basd on Selected Destination Node
62
Start Traversal of Defined Path
64
Toggle out of automated path at any point to manual mode
Original Path Nodes (including Start and Destination) Saved
Create new exit node at point of toggle
66
Explore through manual navigation to new location
68
Choose to rejoin path (from point of exit)
a) Validate original path - checking major dynamic features are still relevant for original path to match user profile options
70
If Path valid
Create Start Node in current location
If Path invalid
Show warning and allow continue or discard original path data
72
Start fresh path search retaing only original destination node
74
76
Create start node at current location
Attach node to network (refer subdivision patent)
88
Perform path search to exit node - to create REJOIN path
Retrieve original path data - discard start to exit, retain exit to destination
80
Combine REJOIN path with original data (exit to destination)
82
Start Traversal of Defined Path
84
Arrive Destination Node
86
Automated Toggle back to manual mode

Figure 16

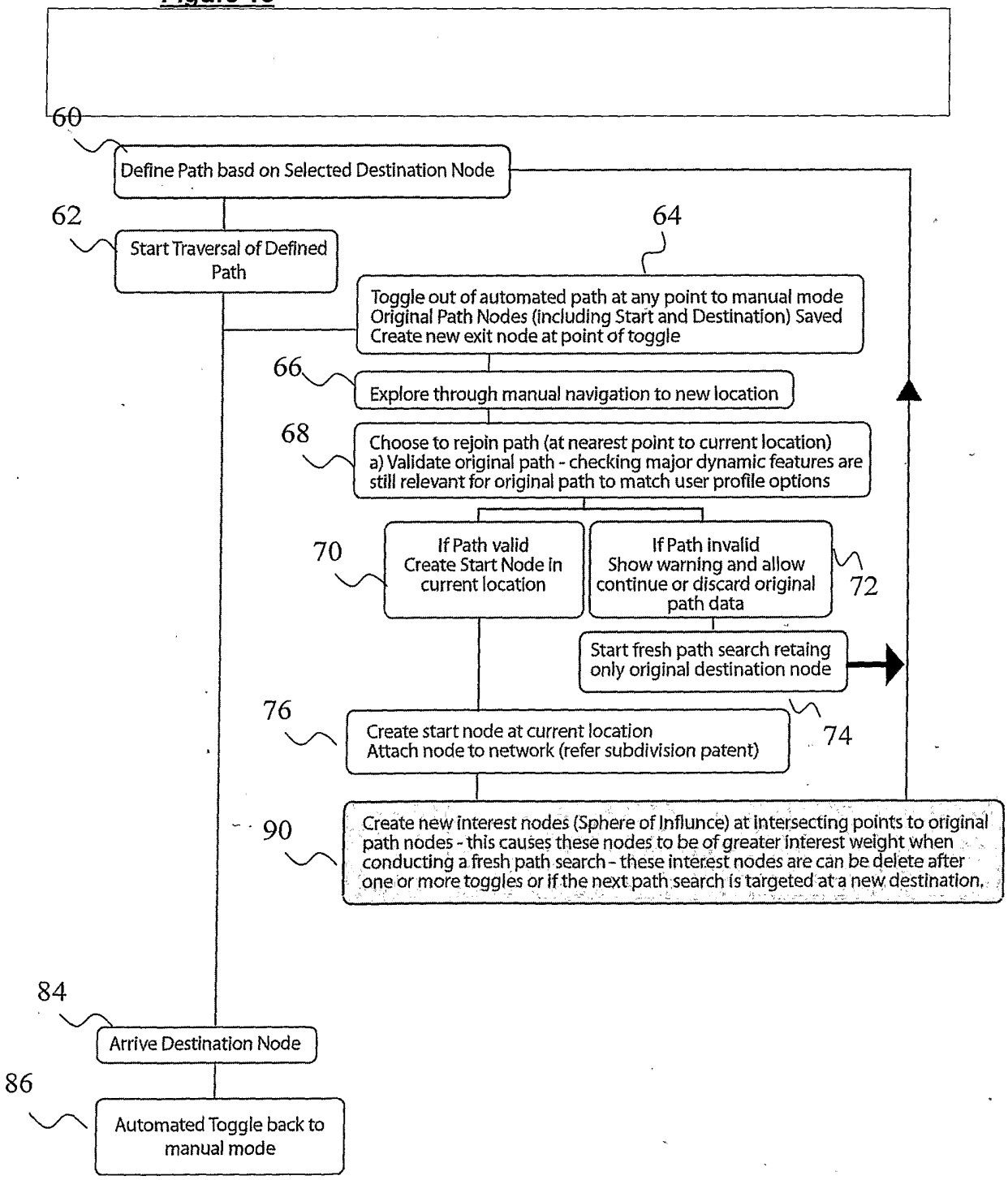

60
Define Path basd on Selected Destination Node
62
Start Traversal of Defined Path
64
Toggle out of automated path at any point to manual mode
Original Path Nodes (including Start and Destination) Saved
Create new exit node at point of toggle
66
Explore through manual navigation to new location
68
Choose to rejoin path (at nearest point to current location)
a) Validate original path - checking major dynamic features are still relevant for original path to match user profile options
70
If Path valid
Create Start Node in current location
If Path invalid
Show warning and allow continue or discard original path data
72
Start fresh path search retaing only original destination node
76
Create start node at current location
Attach node to network (refer subdivision patent)
74
90
Create new interest nodes (Sphere of Influnce) at intersecting points to original path nodes - this causes these nodes to be of greater interest weight when conducting a fresh path search - these interest nodes are can be delete after one or more toggles or if the next path search is targeted at a new destination.
84
Arrive Destination Node
86
Automated Toggle back to manual mode

Figure 17

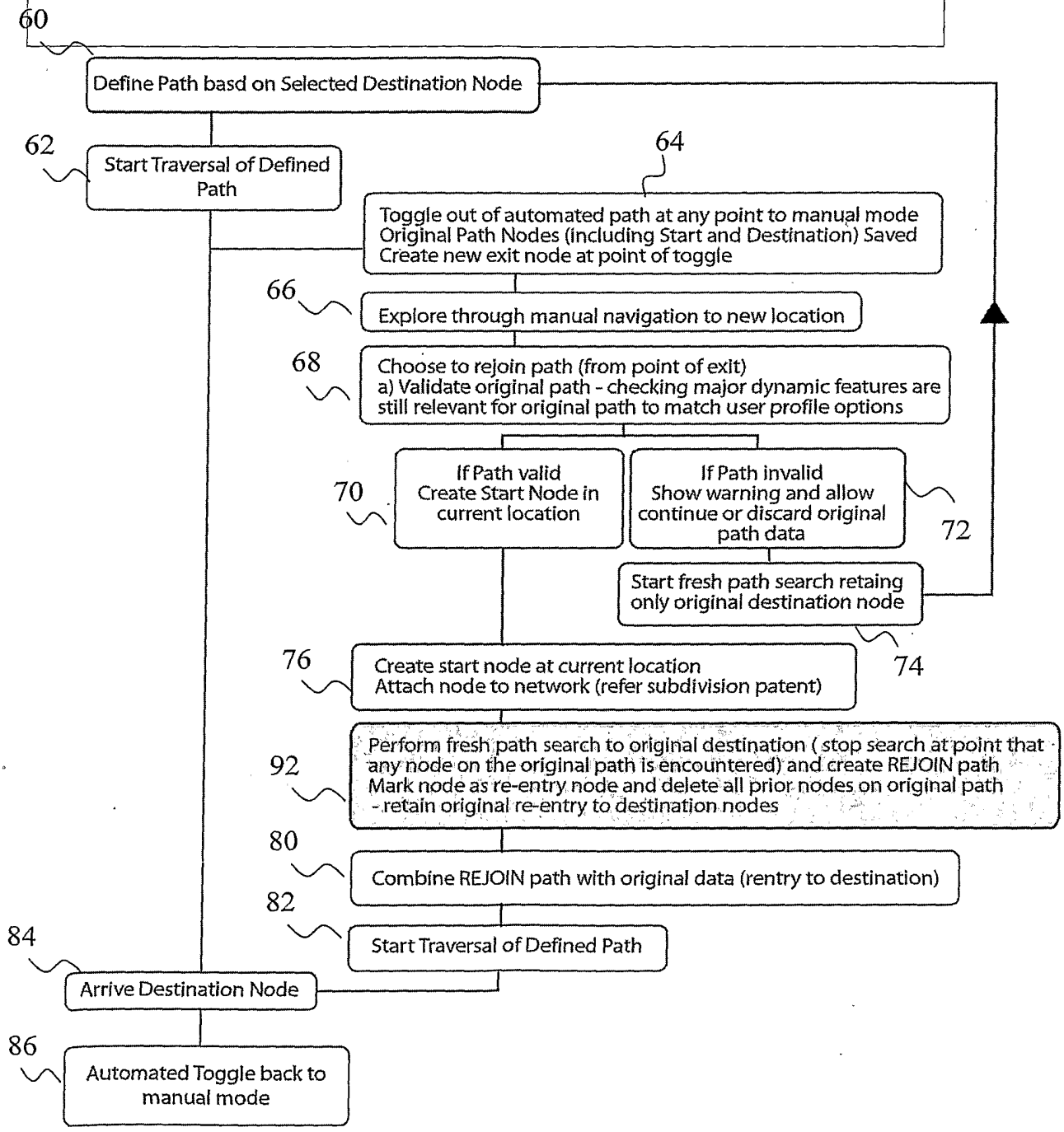

60
Define Path basd on Selected Destination Node
62
Start Traversal of Defined Path
64
Toggle out of automated path at any point to manual mode
Original Path Nodes (including Start and Destination) Saved
Create new exit node at point of toggle
66
Explore through manual navigation to new location
68
Choose to rejoin path (from point of exit)
a) Validate original path - checking major dynamic features are still relevant for original path to match user profile options
70
If Path valid
Create Start Node in current location
If Path invalid
Show warning and allow continue or discard original path data
72
Start fresh path search retaing only original destination node
74
76
Create start node at current location
Attach node to network (refer subdivision patent)
92
Perform fresh path search to original destination (stop search at point that any node on the original path is encountered) and create REJOIN path
Mark node as re-entry node and delete all prior nodes on original path - retain original re-entry to destination nodes
80
Combine REJOIN path with original data (rentry to destination)
82
Start Traversal of Defined Path
84
Arrive Destination Node
86
Automated Toggle back to manual mode

Figure 18

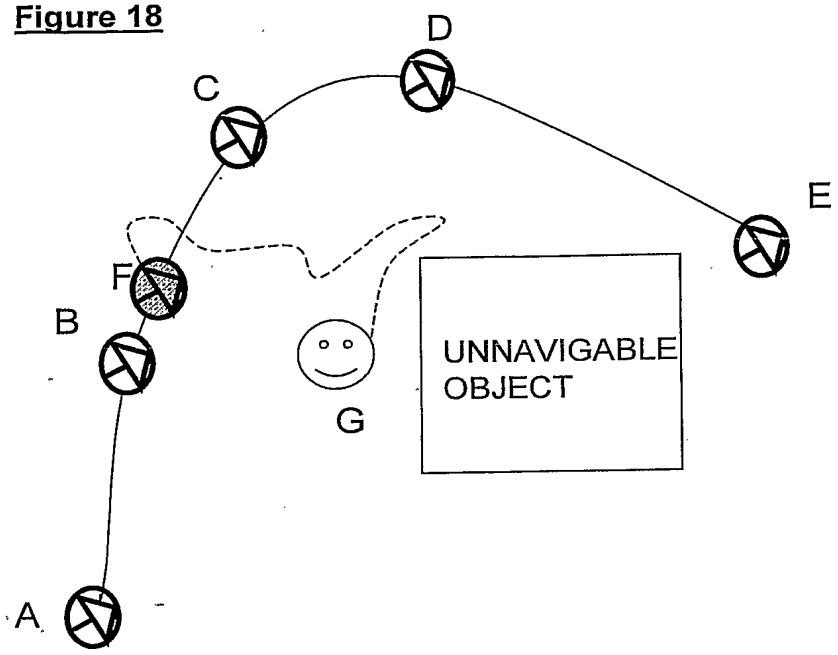
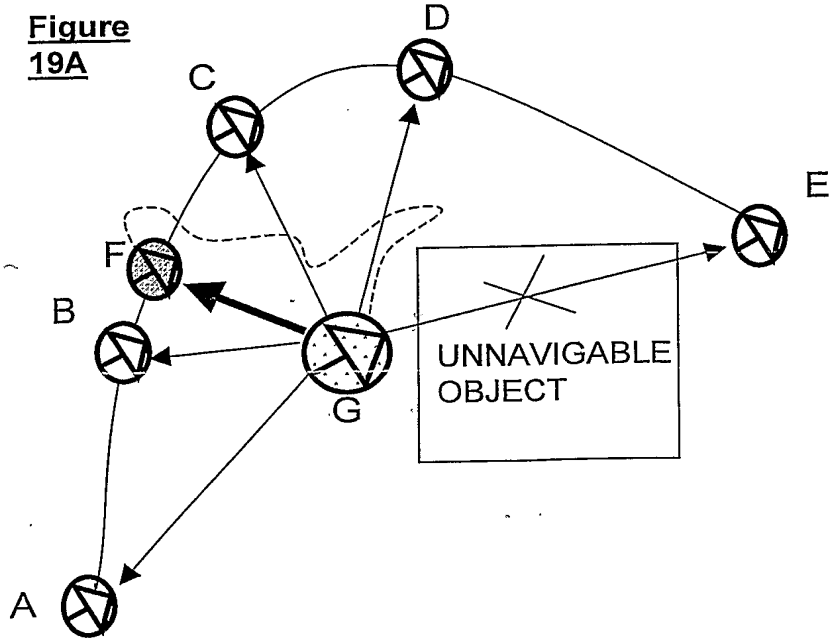


Figure 19A



NAVIGATIONAL NODE

Figure 19B

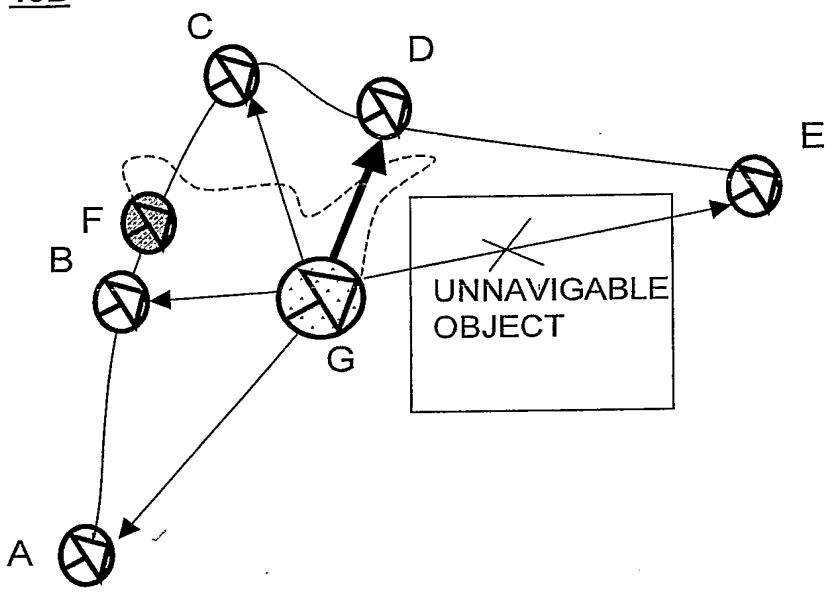


Figure 19C

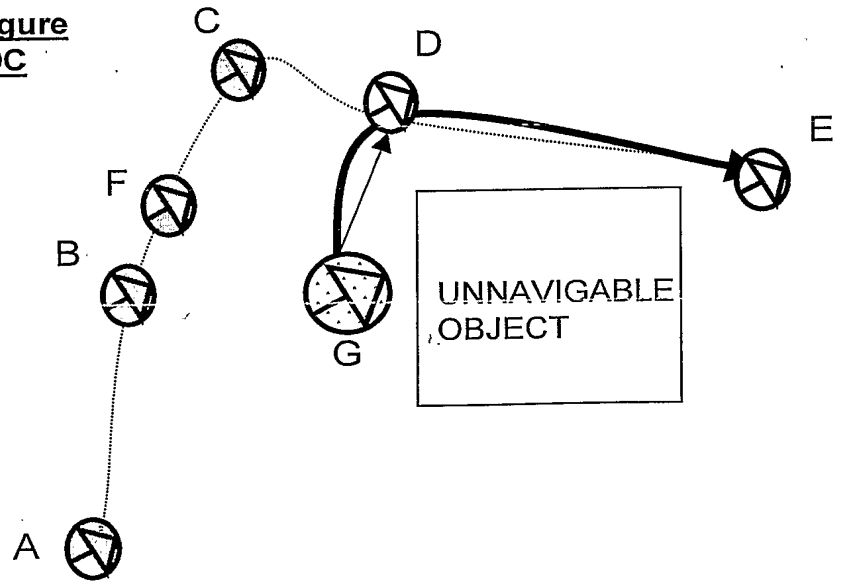


Figure 20A

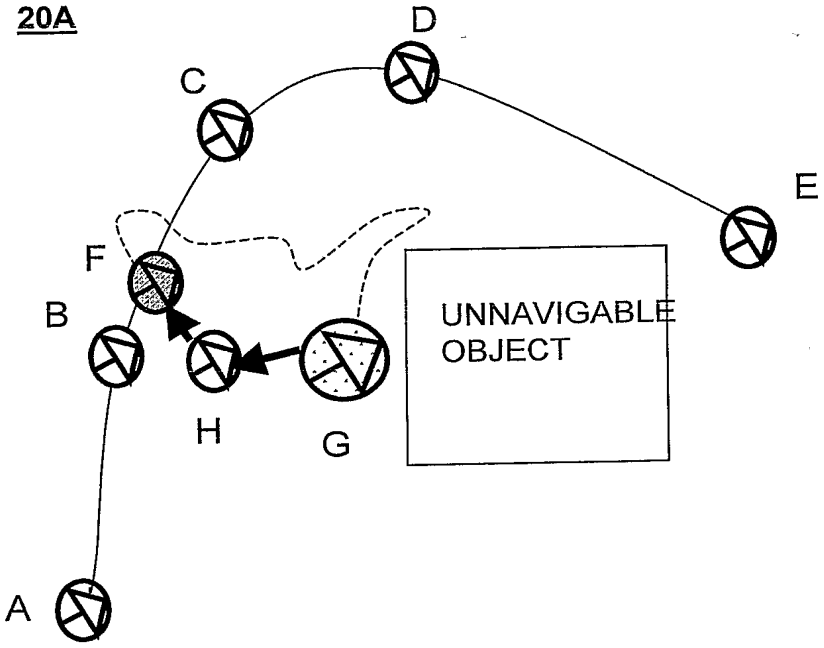


Figure 20B

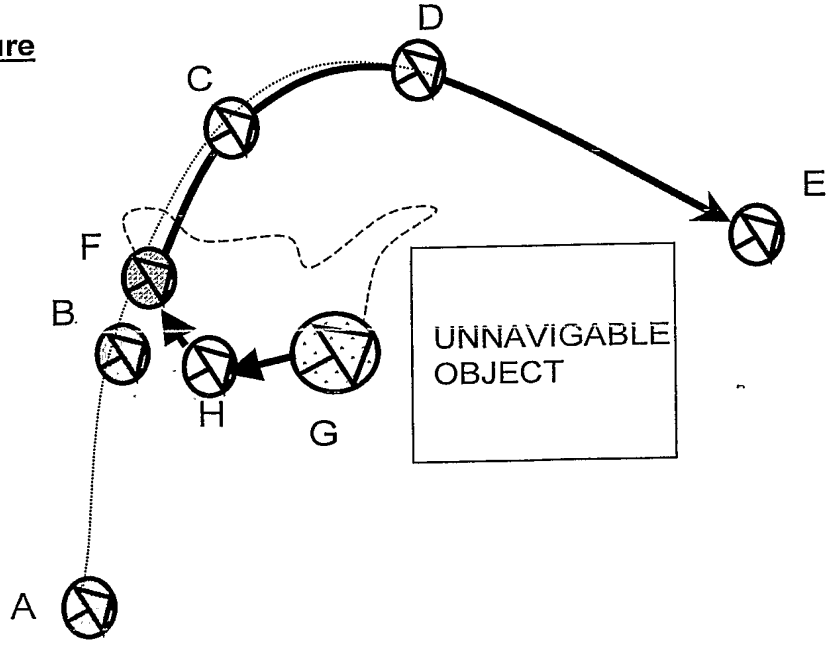


Figure 21

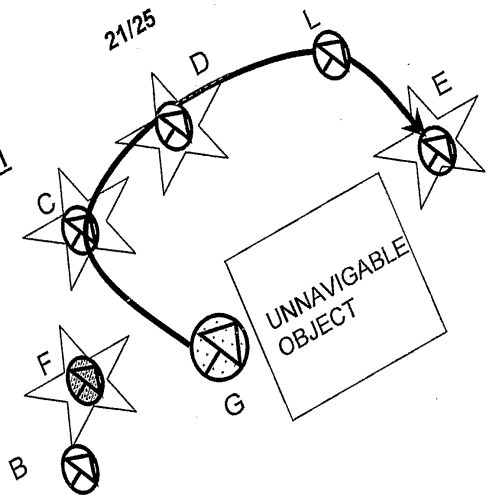


Figure 22

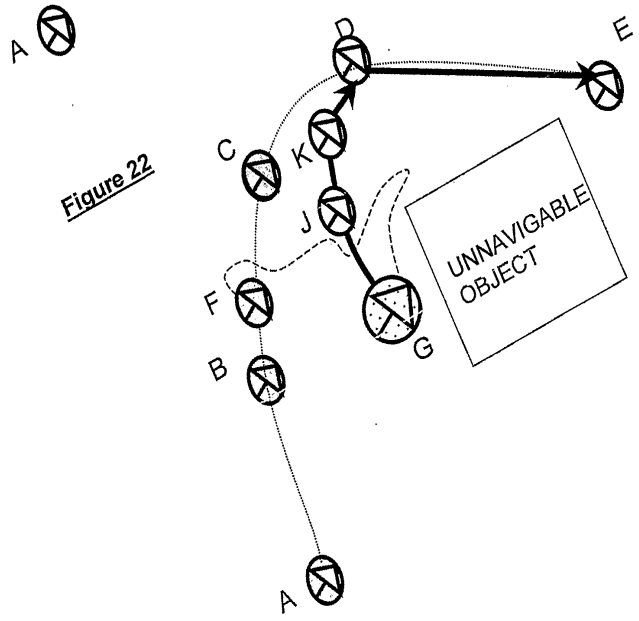


FIGURE 13A

OPTIMUM NODE CONNECTION

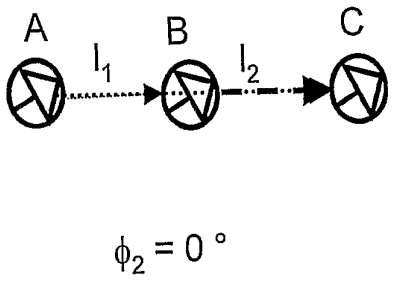


FIGURE 23B

AVAILABLE NODE CONNECTION

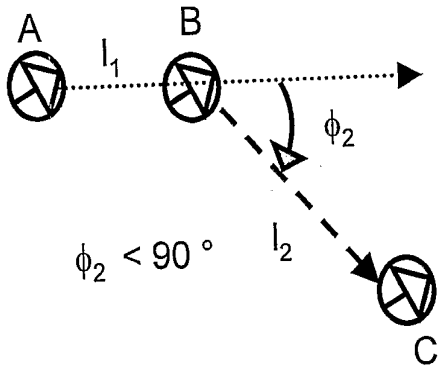
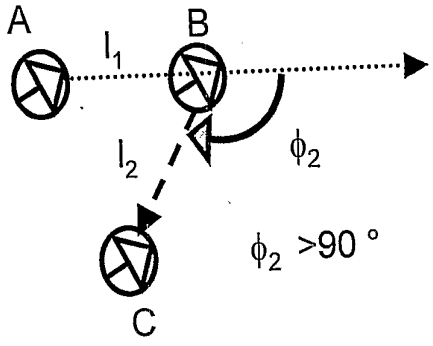


FIGURE 23C

INEFFICIENT NODE CONNECTION



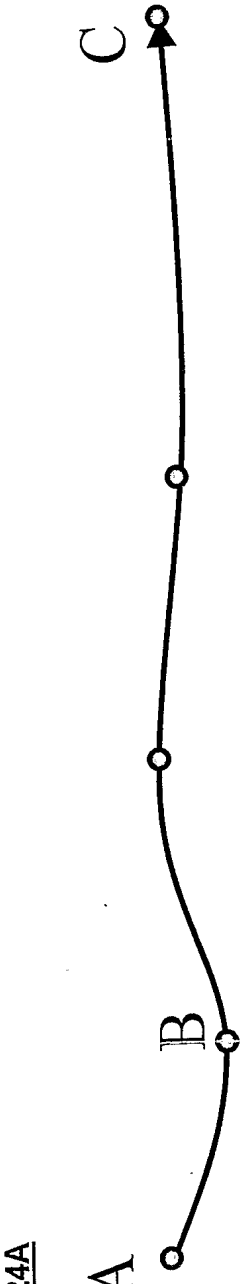


FIGURE 24A

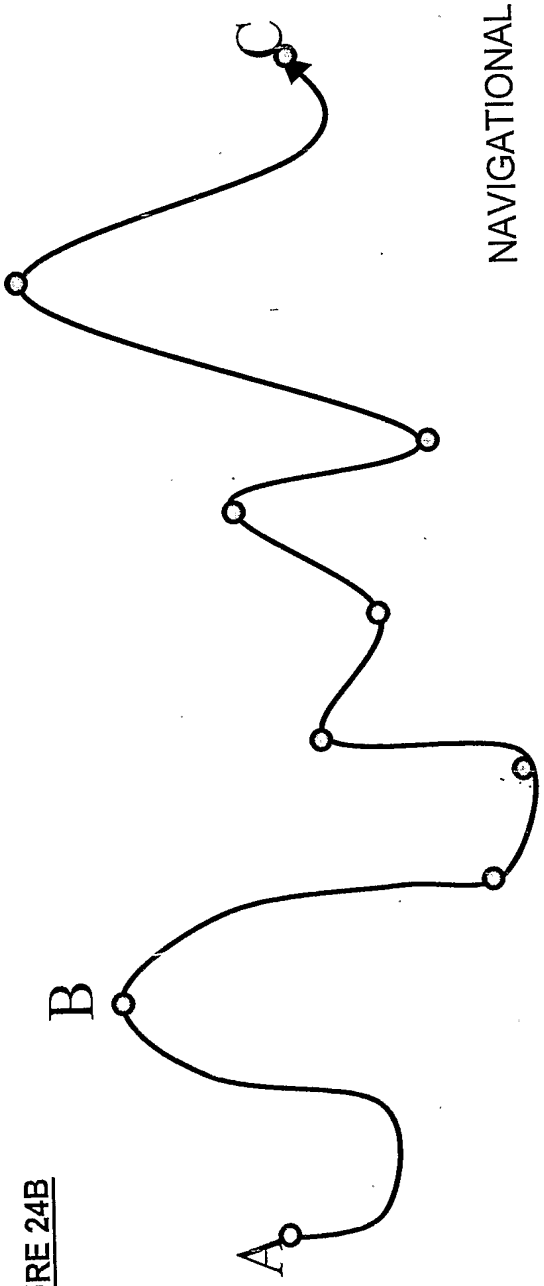


FIGURE 24B

FIGURE 25A

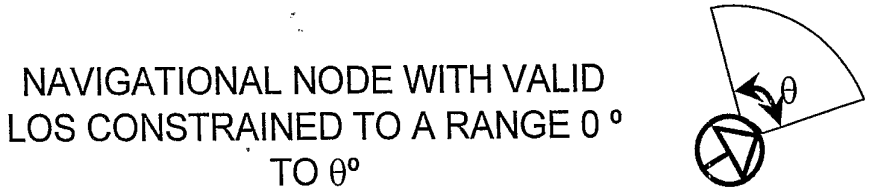
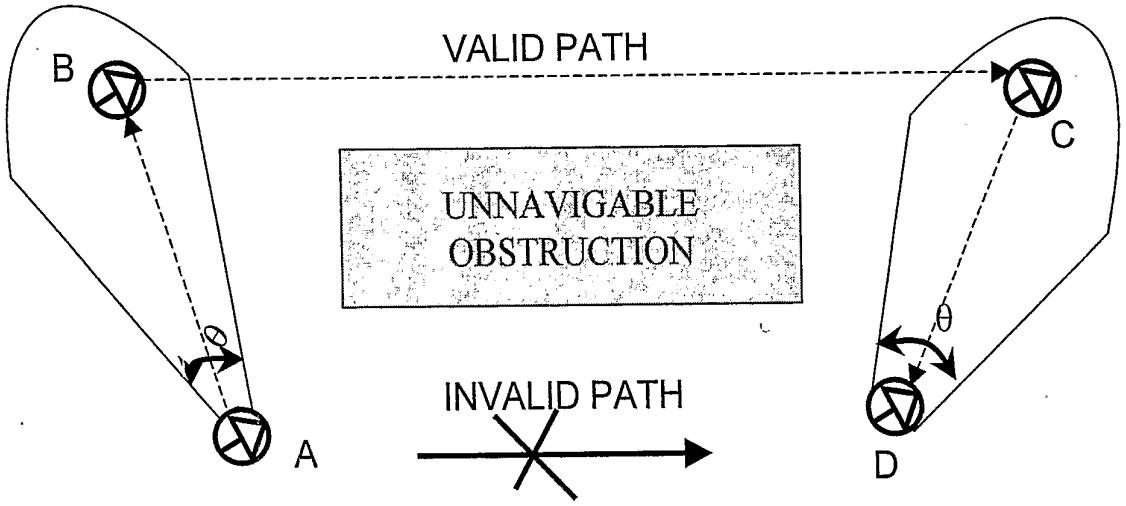


FIGURE 25B



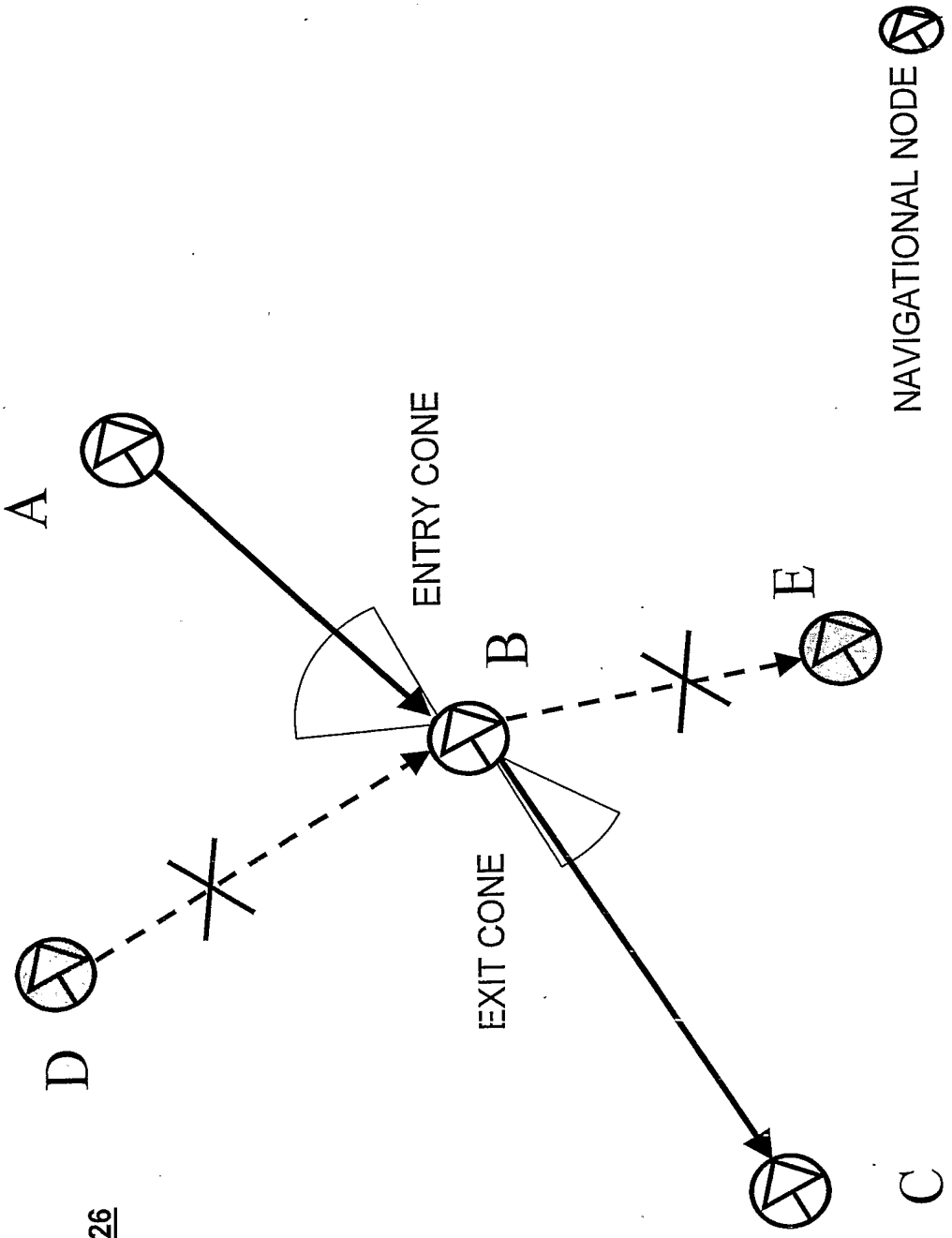


Figure 26